Pramati Server 4.1

Installation and Configuration Guide

Pramati Technologies

Pramati Server 4.1 Installation & Configuration Guide
ID 1230-034
April 2004

Pramati Technologies
301 White House, Begumpet
Hyderabad 500 016
India

info@pramati.com
www.pramati.com

Communications Design Group, Pramati Technologies.
Made in India.

Copyright 2002 Pramati Technologies. All rights reserved. Pramati is a registered trademark of Pramati Technologies in United States and India.

Java and all Java based marks are trademarks or registered trademarks of Sun Microsystems, Inc., in United States and other countries. WebLogic is a registered trademark of BEA Systems, Inc. Informix Cloudscape is a registered trademark of Informix Corporation. Oracle and Oracle 9i are registered trademarks of Oracle Corporation. ANT is a trademark of Apache Software Foundation. Red Hat is a registered trademark of Red Hat, Inc. Netscape is a registered trademark of Netscape Communications Corporation. Opera for Windows is the registered trademark of Opera Software, Inc. AIX is a trademark or registered trademark of IBM Corporation. UNIX is a registered trademark of the Open Group. Apple, Mac, Power Mac and Mac OS are registered trademarks of Apple Computer, Inc. Microsoft, Windows, Windows NT, Windows 2000 and Internet Explorer are Registered Trademarks of Microsoft Corporation. All other trademarks may be the registered trademarks of their respective owners.

No third party intellectual property right liability is assumed with respect to the information contained herein. While the information in this publication is believed to be accurate, Pramati Technologies makes no warranty of any kind to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

Pramati Technologies shall not be liable for errors contained herein, or for incidental or consequential damages in connection with the furnishing, performance or use of this material. Pramati Technologies assumes no responsibility for errors or omissions contained in this book. This publication and features described herein are subject to change without notice. No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, photocopying, recording or otherwise, without prior written consent of Pramati Technologies.

1 Preface 1
Document Conventions 1
Contact Information 1

2 System Requirements for Installing Pramati Server 1
System Requirements 1
Hardware Requirements 1
Operating System Requirements 1
Software Requirements 2
Database Requirements 2
Verifying and Using JDK and JRE 2

3 Getting Server Install File 5
Receiving License Key 6
Upgrading License 6
Upgrading License using Command Shell 6
Upgrading License at Server Startup 6

4 Installing Pramati Server on Unix-based Platforms 9
Verifying Server installation on Unix-based platforms 9
Verifying installation of Deploy Tool on Unix-based platforms 9
Verifying installation of Message Server on Unix-based platforms 10

5 Installing Pramati Server on Windows 13
Installing in non-GUI mode from command line 13
Verifying Server installation 14
Verifying installation of Deploy Tool 14
Verifying installation of Message Server 15

6 Silent Installation of Pramati Server Using XML 17

7 Setting up Pramati Server as a Service 19
Setting up Pramati Server as a Service on Windows 19
Setting up Pramati Server as a Unix Daemon 20
Installing Java Wrapper 21
Configuring more than one Pramati Server Nodes 22
Customizing Default Parameters 22
Configuring JVM Parameters 23
Configuring Classpath Parameters 24
Troubleshooting Tips 24

8 Pramati Server Components 25
Configurable Server Components 25
Pramati Server Standalone node 25
Pramati Server Cluster node 25
Pramati Server Cluster EJB node 26
Pramati Server Cluster Web node 26
Pramati Server Cluster BOTH node 26
Pramati Embedded Message Server 26
Pramati Server Standalone Message Server node 26
Pramati Server Load Balancer 26
Pramati Server Administration Service 27

Pramati Server Management Console 27
Java Client 27
Third Party Integration with Apache/IIS 27
Configuring Components using Pramati Server Administration Service 27
Configuring Components using Console 27
Configuring Components by Modifying Configuration XMLs 27

9 Adding a User and Changing Password 31
Adding User 31
Deleting User 32
Changing Password 32
Using User Settings 32
Using Security option 32

10 Introduction to Pramati Server Administration Service 35
Role of Administration Service 35
How does Server Administration Service work 36
Starting Server Administration Service 36

11 Configuring Standalone Pramati Server Using Admin Service 37
Using Default Server 37
Starting the Default Server on Windows 37
Starting the Default Server on Unix 37
Configuring Standalone Server 37
Creating Standalone Server 38
Viewing details of added Standalone Server 39
Configuring Standalone Server 39
Starting Standalone Server 39
Connecting to Standalone Server 40
Stopping Standalone Server 40
Removing Standalone Server 40

12 Adding Web Load Balancer using Administration Service 43
Load Balancing Scenarios 43
Adding a Load Balancer node 45
Configuring Load Balancer 46
Viewing Load Balancer details 47
Starting Load Balancer 47
Connecting to Load Balancer 48
Stopping Load Balancer 48
Removing a Load Balancer 48

13 Configuring Web Load Balancer Using XML 49

14 Configuring J2EE Cluster Using Server Administration Service 51
Configuring J2EE Cluster 51
Modifying Cluster 52
Starting Cluster 53
Connecting to Cluster 53
Deploying an application on Cluster node 53
Effects of deploying an application on Cluster node 54
Stopping Cluster 54

Deleting Cluster....54

15 Configuring Message Server as a Standalone Node....55
Configuring Standalone Message Server....55
Creating Standalone Message Server....55
Configuring Standalone Message Server....55
Starting Standalone Message Server....56
Viewing details of Standalone Message Server....56
Connecting to Standalone Message Server....56
Stopping Standalone Message Server....57
Removing Standalone Message Server....57
Using Loaded Send Queues....57
Configuring the Message Server for Loaded Queues....57
Starting Standalone Message Server using Command line....58
Understanding Message Server Configuration....59
Configuring Large Message Handling....59
Configuring JMS Persistence Framework....59

16 Starting a Server Node Using Command line....63

17 Configuring Message Server Cluster Using Server Admin Service65
Configuring Message Server Cluster....65
Adding node(s) to Message Server Cluster....66
Starting Message Server Cluster....67
Connecting to Cluster....67
Viewing Message Server node details....68
Modifying Message Server Cluster configuration....68
Stopping Message Server node....68
Stopping Cluster....68
Deleting Cluster....69
Running client for Message Server Cluster....69
Managing Cluster Nodes....69

18 Creating and Configuring Pramati Server Nodes Using XML....71
Overview....71
Understanding Pramati Server nodes....71
Key terminology....71
Pramati Server configuration files....72
Pramati Server instances that can be configured....73
Standalone Server configurations derived from an instance....73
Server Cluster Node configurations derived from an instance....73
Structure of Nodes Directory....74
The installation directory structure....74
Server Instance Directory Structure....74
Procedure for creating Pramati Server Instances....75
Configuration File Dependency....76
Creating a Standalone Server....76
Enabling embedded Message Server in a Standalone Server....78
Restricting services in a Standalone J2EE Server....78
Creating Standalone Message Server....79
Customizing Server Footprint....79

19 Using Pramati Server NodeCreator 81
Using NodeCreator with Command line 81
Using NodeCreator with XMLs 82
Using - configuration flag for custom node configuration 84
Cluster Configuration using NodeCreator 85
Usage Objective 85
Configuring J2EE Server Cluster using NodeCreator 86
Creating LB Node using NodeCreator 87

20 Creating Pramati Server Clusters Using XMLs 89
Cluster Definition 89
Uploading Configuration to DB when DB is used for Configuration Persistence 90
Creating a Cluster Instance 90
Setting up a J2EE Cluster 92
J2EE Cluster Node Types 93
Advanced Configuration 94
Setting up Web Replication and Persistence 94
Setting up EJB Persistence 95
Updating cluster configuration using XML (when DB is used for configuration) 95
Configuring J2EE Server Cluster using XMLs 96
Deploying an application on Cluster node 97
Configuring J2EE Server Cluster using NodeCreator 97

21 Configuring Message Server Cluster Using XML 99
Configuring Message Server Cluster using XML 99
Setting up an HA Message Server Cluster 100
Starting Message Server node from Command line 101

22 Configuring JMS Adapters in J2EE Servers 103
Sample Configuration for JMS Adapters 103

23 Configuring Pramati Server For Firewalls 105
HTTP tunneling (Port Filtering firewalls) 105
Configuring client 105
Configuring RMI Server 106
Working with NAT firewalls 106
Default Socket Factory 107
Configuring Server for firewalls 107
Hosting Scenarios 108
All installations on the host machine 108
Pramati Web Container in DMZ 108
Third-party HTTP server configured with Pramati Server 108
Third-party HTTP server and Servlet engine 108

24 Upgrading Pramati Server License 109
Reading the Current License Information 109
Upgrading license 109
Upgrading License at Server Startup 110

25 Migrating from Pramati Server 3.0 to Pramati Server 4.1 111
Starting the Servers 111
Starting J2EE Server 111

Starting JMS Server....112
Starting Web Load Balancer....112
Offline preparation of client.jar....112
RemoteShell....112
Command Line Operations....112
Shell Commands....113
System Properties....113
Starting RemoteShell....114
List of all start up options to start Pramati Server....114
New Server Configuration XMLs in Version 4.1....115
New Service Configurators and other XMLs in Version 4.1....115
New configuration tags in deploy-config.xml....115
Handling of RARs....115

26 Downloading and Installing WebStart....117
Downloading and Installing WebStart on Windows....117
Downloading and Installing WebStart on Solaris and Linux....117
Server-side Setup of WebStart....117
Modifying Server Configuration for WebStart....118

27 Pramati Server Communication Ports and Protocols....119
Ports used by Server Administration Service....119
Ports used by Pramati Server Standalone Node....120
Ports used by Pramati Server Cluster Node....120

28 Configuration Layouts....121
Pramati Web Server in DMZ....122
Third-party HTTP server replaces Pramati HTTP Server....122
Third-party HTTP server + servlet engine....123
Independent Pramati Servers on the network....123

Preface

The Installation and Configuration Guide to Pramati Server 4.1 provides you with complete and detailed instructions for installing and configuring Pramati Server. It also introduces you to the important feature of the Server.

Pramati Server installs with a Deploy Tool, the Console, and an embedded Message Server.

Document Conventions

The following conventions are commonly used in this guide.

Table 1: Document conventions used in the Administration Guide

Convention	Used in place of
Server	Pramati Server
Console	Pramati Server Management Console
Item A > Item B	Signifies menu and sub-menu selections
`code`	The font used for writing code snippets
Note:	Indented paragraphs in italics denote notes
<install_dir>	Server installation directory

Contact Information

Detailed information on this release of the product is given in the Release Notes. For additional help with using Server and troubleshooting, visit http://www.pramati.com or visit http://www.pramati.com/forum.

System Requirements for Installing Pramati Server

This document provides information about system requirements for installing and configuring Pramati Server 3.5 on Windows and Unix platforms.

System Requirements

Before you start installing Pramati Server, check for:

- Hardware Requirements
- Operating System Requirements
- Software Requirements
- Database Requirements

Hardware Requirements

The hardware requirements are:

- Any hardware platform that supports Java with minimum 128 MB RAM (256 recommended).
- Minimum 100 MB hard disk space during installation. Also, make sure that the temporary directory set by the Environment Variable TMP has a minimum disk space of 100 MB.
- Minimum 80 MB of free disk space for downloading the installation file from the web site. This is not applicable if you are installing from the CD.

Operating System Requirements

You can install Server on any platform that supports JDK 1.3.1_01 or above.

The recommended platforms for installing Pramati Server 4.1 on Windows are:

- Windows 2000™ Professional and Server
- Windows NT™ 4.0 with Service Pack 5 or higher
- Windows XP Professional

The recommended platforms for installing Pramati Server 4.1 on Unix based platforms are:

- RedHat Linux™ 6.2
- RedHat Advanced Server 2.1
- Caldera OpenLinux Server 3.1.1
- Sun Solaris™ 7 and 8

- HP-UX 11i

Software Requirements

The JDK and Browser requirements are provided in the following sections.

JDK

Pramati Server has been tested on JDK 1.3.1_01 and above, including JDK1.4.

Browser

Pramati Server can be administered over the web using the Pramati Server Management Console. The Console has been tested with the following browsers that support HTML 3.2:

- Internet Explorer 5.5 or above
- Netscape 4.7 or above
- Opera 5.1
- Mozilla

Database Requirements

Pramati Server is built to the J2EE platform specification and works with any RDBMS that supports JDBC 2.0 standard extensions. The Server has been tested with the following database servers:

Table 2: Databases supported and Driver versions

Database	JDBC Driver
Oracle 8i	Oracle Thin Driver Version 8.1.6.0.1, Merant DataDirect Oracle Driver 2.2
Oracle 9i	Oracle Thin Driver Version 8.1.7
Microsoft SQL Server 7.0	Merant DataDirect SQL Server Driver 2.2, JTurbo SQL Server Driver 2.0, i-Net Sprinta 4.15 SQL Server Driver
MySQL Max 3.23.48	mm.mysql 2.0.4
Informix 7.3.0 TC3	Informix JDBC Driver for Informix Dynamic Server 2.10.JC1N361, Merant DataDirect Informix Driver 2.2
Cloudscape 4.0	Cloudscape 4.0 JDBC Driver 4.0, Cloudscape 4.0 RMI JDBC Driver 4.0
MySQL 3.28 with InnoDB tables	MySQL 2.0.4 driver
Sybase 12.5	jConnect for JDBC 5.5

The *Release Notes* contain the complete and latest platform certification report and is available on the website (www.pramati.com) or along with the Server installation.

Verifying and Using JDK and JRE

To verify the JDK version on your machine, enter `java -version` at the command prompt. The value that is returned is the JDK version on your machine.

Using JDK

If the J2EE application has JSP pages, which need to be compiled at runtime, JDK may be required on the host. This is a common scenario where only the deployment mapping XMLs are required to be in the EAR. If the application provider wants to permit re-mapping or modification of deployment information at the customer site, then JDK must be installed on the host machine.

Using JRE

If a customer of the application wants to use JRE, pre-compiled generated code must be shipped by the application provider. This can be done in the form of a prepared archive (.par file) which is readily deployable on Pramati Server. Note that when making a .par file, Pramati Server Deploy Tool does not precompile the generated code by default. Also, when JRE is used, deployment information cannot be modified at the customer site.

Getting Server Install File

Pramati Server can be downloaded as a binary from http://www.pramati.com/products. If you purchase a license with media and documentation, you will receive the product on a product distribution CD along with printed documentation.

The Server is available as an executable (EXE) file for Windows users, and as a Java archive (JAR) file for Unix users. The distribution has the following components:

- Pramati Server (EJB and Web Container, Pramati HTTP Server, embedded Message Server)
- Pramati Deploy Tool
- Pramati Server Management Console
- Pramati Load Balancer

Additionally, if you are using a CD, you will get separate installables for:

- Pramati Deploy Tool
- Pramati Message Server (all features of Embedded Message Server plus high-availability solution)

Documentation is distributed on the CD. For more information on the distribution, refer to the accompanying *Release Notes*.:

Table 3: List of components contained in the Pramati Server distribution

Server Component	Distribution	Description
Pramati EJB 2.0 Server	Integrated	Deploy EJB 2.0 components
Pramati Web Server	Integrated	Serve HTML, JSP 1.2 and Servlet 2.3 components
Pramati Administration Service	Integrated	Discovery and configuration of Servers
Pramati Server Management Console	Integrated	Web-based management of Servers
Pramati Deploy Tool	Integrated. Also available as a separate installation from product CD.	Deploy J2EE 1.3 applications
Embedded Message Server	Integrated. Also available as a separate installation from product CD.	Run messaging applications using JMS 1.0.2

Receiving License Key

Normally, after the registration process is complete and the file is downloaded from www.pramati.com, an evaluation license key is send to the e-mail address specified by you during registration.

If you have purchased the product and are installing from a product distribution CD, the license key is provided with the CD package in a printed license certificate.

If you have trouble with using the license key, send a mail to `license@pramati.com`.

Upgrading License

License can be upgraded using the following two methods:

- Using the command shell
- At Server Startup

Upgrading License using Command Shell

To upgrade the license using the command shell, use the `upgrade_license` command. The following steps are interactive. See below for a sample license upgrade:

```
j2eeadmin@default> upgrade_license
Please Enter the following values required by the command:
Note: Fields followed by * are mandatory
You can press 'Enter' for default values shown in [ ]
Typing 'q' and pressing enter at any point will cancel the current command
j2eeadmin@default>  - LicenseKey * #XXXXXXXXXXXXXXXX
License upgraded successfully
Upgraded license successfully. New license details are
License Information:
Licensee            : CustomerName
Installation Date   : 1 Mar 2004
Expiry Date         : 1 Mar 2006
License Type        : Development
Product             : Pramati Server 4.1
Start Date          : --
Available Time      : --
j2eeadmin@default>
```

Upgrading License at Server Startup

License can also be upgraded at server startup by using the -upgradelicense option as shown in the example below:

```
> runserver -upgradelicense XXXXXXXXXXXXXXXX
```

```
C:\PServer41_1127\server\bin>java.exe -Din-
stall.root=C:\PServer41_1127\server -Djava.security.pol-
icy=C:\PServer41_1127\s
pramati\pramati.java.policy -Djava.security.auth.pol-
icy=C:\PServer41_1127\server\lib\pramati\pramati.jaas.policy -Dorg.x
ser=org.xml.sax.helpers.XMLReaderAdapter -
Dorg.xml.sax.driver=org.apache.crimson.parser.XMLReaderImpl
com.pramati.Server -upgradelicense 08xEW0AJDCsCFEQ6
```

License upgraded successfully

Note: *Entering a wrong license upgrade key displays an error message. Please check the license key before entering the details.*

Installing Pramati Server on Unix-based Platforms

To install Pramati Server on Unix based platforms:

1. Verify your existing system configuration against the requirements for Pramati Server.
2. Execute `java -jar PServer41.jar` from the directory where the installation file is downloaded (or from the CD, if you are installing from a product distribution CD).
3. The installation wizard starts and the license agreement appears. To proceed with the installation, select "I accept the terms of license agreement" and click 'Next'.
4. Select the directory where the Server is to be installed. You *must* have write permissions on this directory.
5. Specify the path of the supported JDK (say, JDK 1.3.1_01) installed on the machine. Specifying an incorrect path leads to an incomplete installation.
6. Enter the licensee name. This can be the name of the evaluator or the customer in whose name the license has been issued.
7. Enter the license key.

The installation starts using the above values.

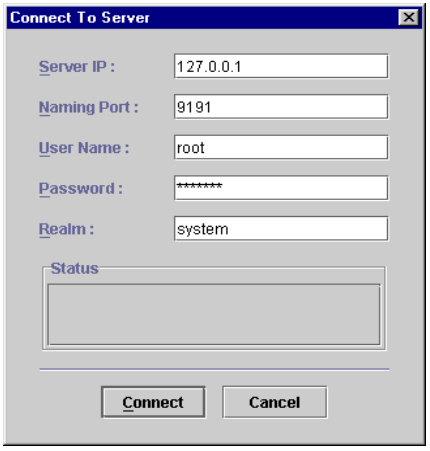
Verifying Server installation on Unix-based platforms

1. Start the shell and go to the directory <install_dir>/server/bin.
2. Make `runserver.sh` executable by typing the command:
   ```
   $ chmod 755 runserver.sh.
   To run the executable file, type: ./runserver.sh
   ```
3. This should start the Server and bring up the Server Shell prompt. This prompt is used for performing Server administration and deployment tasks.
4. To shut down Server, run `shutdown` at the default prompt. The Server should stop and return you to the shell prompt.

Verifying installation of Deploy Tool on Unix-based platforms

1. To verify installation of Deploy Tool, Pramati Server should be started.
2. To verify the Deploy Tool in a Pramati Server installation, open a command prompt and go to the directory `<install_dir>/server/bin`.
3. To verify a Deploy Tool installation, go to the directory, `<install_dir>/deploytool/bin`.

4 Execute `rundeploytool.sh` at the command prompt. This opens the 'Connect to Server' dialog box.



5 Provide the following information:
- Server IP: Specify the Server IP or the host name where the server is running. For example, 192.168.1.36.
- Naming Port: Specify the port on which the server is listening. For example, 9191
- User Name: Specify the username provided while configuring the realm. For the realm *system*, the username is *root*
- Password: Specify the password provided while configuring the realm. For the realm *system*, the password is *pramati*
- Realm: Specify the realm name. The default value is *system*

6 Click 'Connect' or press <enter> to open the Deploy Tool.

To exit the Deploy Tool, select Archive > Exit. You can also use the 'Exit' button provided.

Verifying installation of Message Server on Unix-based platforms

1 If you are verifying Embedded Message Server, open a command prompt and go to the directory `<install_dir>/server/bin`. Execute `runserver -startjms` command.

2 If you are verifying the standalone Pramati Message Server, open a command prompt and go to the directory `<install_dir>/jms/bin`. Execute `runjms.sh` at the command prompt.

3 This should start the Message Server and bring up the Server Shell prompt. This prompt is used for performing Message Server administration and deployment tasks.

4 To shut down the Message Server, run `shutdown` at the default prompt. The Message Server should stop and return you to the command prompt.

To install Pramati Server on Windows:

5 Verify your existing system configuration against the requirements for Pramati Server.

6 Execute `PServer41.exe` from the directory where the installation file is downloaded (or from the CD, if you are installing from a product distribution CD).

7 The installation wizard starts and the license agreement appears. Read the license agreement. To proceed with the installation, check "I accept the terms of license agreement" and click 'Next'.

8 Select the components to install and name the directory where they should be installed.

9 Specify the path of the supported JDK (say, JDK 1.3.1_01) installed on the machine. Specifying an incorrect path leads to an installation failure.

10 Enter the licensee name. This can be the name of the evaluator or the customer in whose name the license has been issued.

11 Enter the license key.

The installation starts using the above values. A successful installation will install all the specified components under the specified installation directory. Shortcuts are created under the Start > Programs menu.

Installing in non-GUI mode from command line

Server can be installed in an interactive command line mode, instead of the GUI-based installation wizard. This is possible only with the JAR distribution, `PServer41.jar`, of the Server, which is available with the installation CD. The JAR distribution installs on Windows as well as Unix. Run `java -jar PServer41.jar -cmdLine` at the command prompt and enter these values as the command line prompts:

Table 4: Command line entries for installing Pramati Server in a non-GUI mode

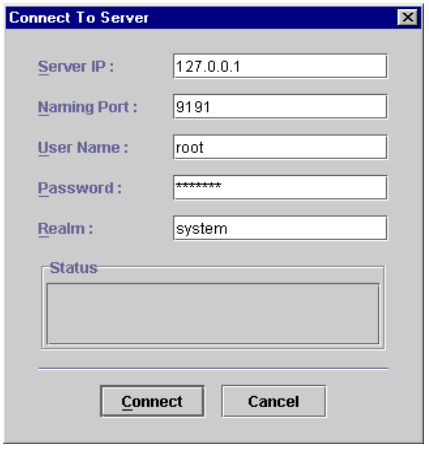
Prompt	Value
Name	Enter a user name
Key	Enter the valid license key
Installation Directory	Enter the directory where you want to install Server
Java Home	Specify the path of supported JDK (say, JDK 1.3.1_01) installed on the machine. Specifying an incorrect path will lead to an installation failure.

Verifying Server installation

1. Open the command prompt and go to the directory <install_dir>/server/bin.
2. Run `runserver.bat`.
3. This starts the Server and brings up the `j2eeadmin@default:>` prompt. This prompt is used for performing Server administration and deployment tasks.
4. To shut down Server, type `shutdown` at the default prompt. The Server stops, and you return to the command prompt.

Verifying installation of Deploy Tool

1. To verify installation of Deploy Tool, Pramati Server should be started.
2. Open a command prompt and go to the directory <install_dir>/server/bin.
3. If you are verifying a Deploy Tool installation, go to the directory <install_dir>/deploytool/bin.
4. Execute `rundeploytool.bat` at the command prompt. This opens the 'Connect to Server' dialog box.



5. Provide the following information:
   - Server IP: Specify the Server IP or the host name where the server is running. For example, 127.0.0.1
   - Naming Port: Specify the port on which the server is listening. For example, 9191
   - User Name: Specify the username provided while configuring the realm. For the realm *system*, the username is *root*
   - Password: Specify the password provided while configuring the realm. For the realm *system*, the password is *pramati*
   - Realm: Specify the realm name. The default value is *system*
6. Click 'Connect' or press '<enter> to open the Deploy Tool.

To exit the Deploy Tool, select Archive > Exit. You can also use the 'Exit' button provided.

Verifying installation of Message Server

1. If you are verifying Embedded Message Server, open a command prompt and go to the directory `<install_dir>/server/bin`. Execute `runserver` command.
2. If you are verifying the Standalone Pramati Message Server, open a command prompt and go to the directory `<install_dir>/jms/bin`. Execute `runjms.bat` at the command prompt.
3. This should start the Message Server and bring up the Server Shell prompt. This prompt is used for performing Message Server administration and deployment tasks.
4. To shut down the Message Server, run `shutdown` at the default prompt. The Message Server should stop and return you to the command prompt.

Silent Installation of Pramati Server Using XML

6

To install the server silently, save the file `PServer41.jar` on your local machine on both Windows and Unix. Silent installation is especially useful in scenarios where the Server is being installed inside another application, and shipped to end users.

There are two steps involved in silently installing Server:

1. Generating an XML file (using the `PServer41.jar` or manually)
2. Using the generated file to silently install Server

Follow the given instructions to generate the XML file using the `PServer41.jar`:

1. Verify your existing system configuration against the requirements for Pramati Server described in '*System Requirements for Installing Pramati Server*'.
2. Execute `java -jar PServer41.jar` from the directory where the installation file is downloaded (or from the CD, if you are installing from a product distribution CD).
3. To generate the XML file manually, copy the given sample lines of code in the template XML file, edit it with your values, and then save it in the current directory:

```
<AutomatedInstallation langpack="eng">
     <com.pramati.action.CDKeyPanelAction>
          <name>Mike</name>
          <value>111111111111111</value>
     </com.pramati.action.CDKeyPanelAction>
     <com.pramati.action.TargetPanelAction>
          <installpath>c:\Pramati_Server</installpath>
          <javaHome>c:\jdk1.3.1</javaHome>
     </com.pramati.action.TargetPanelAction>
</AutomatedInstallation>
```

After generating the XML file, Server can be installed silently using this XML file. To install Server silently, follow the given instructions:

1. Enter `java -jar PServer41.jar -silent stu_template.xml` at the command prompt. Here stu_template is the name of the XML file, which was generated earlier.
2. Pressing the <enter> key runs the automated installation with all the default values in the XML file, and the user is not prompted for any details.
3. To get help on the commands that can be used to install Server, enter `java -jar PServer41.jar -help` at the command prompt.

Setting up Pramati Server as a Service

Pramati Server can be installed as a service on Windows or as a daemon on Unix based systems. This helps the administrators in ensuring that the server is always available on system start. Pramati Server uses the Java Services Wrapper (version 3.0.5) to install Server as service on Windows or a daemon in Unix based systems. Java Services Wrapper is an open source tool, which eases the process of installing any Java application as a service by moving all JVM configuration into a platform independent configuration file. More information on Java Services wrapper is available at http://wrapper.tanukisoftware.org/.

All necessary files for setting up Pramati Server as a service are present in the `<install_dir>/services/` directory.

Setting up Pramati Server as a Service on Windows

In this section we will discuss how to set up the Default Server as service on Windows. Make sure you have all the files specified in the table are available before proceeding any further:

File	Description
\bin\license.txt	A text file describing the copyrights of the use of the Wrapper Service. Provided by TanukiSoftware.org.
\bin\readme.txt	A text file describing the instructions for installing the Wrapper Services.
\bin\setup.bat	This file sets the server install path to an environment variable
\bin\Wrapper.dll	A library file
\bin\Wrapper.exe	This is an executable file that is invoked to install the Java application as a service
\bin\wrapper.jar	The Java Services Wrapper classes are present in this executable JAR file
\j2eeserver\pramati_server_default.conf	A configuration file used to launch the server with pre-defined startup parameters like setting classpath, JVM options etc
\j2eeserver\install_server_default.bat	A batch file used to install the service on Windows
\j2eeserver\remove_server_defaultbat	A batch script to remove the default service

To install the server as a service:

1. Run `setup.bat` script present in the `<install_dir>/services/bin/` directory. This will set up the necessary environment to install the server as a service.
2. Change directory to `<install_dir>/services/j2eeserver/` and run `install_server_default.bat`. This will install Pramati Server at node `default` as a service. Make sure the default node is created before executing this setup.

```
> install_server_default.bat
wrapper  | Pramati J2eeServer default installed.
```

To start the service, use the Windows `net start` command which displays messages as shown below:

```
> net start PramatiServerdefault
The Pramati J2eeServer default service is starting....
The Pramati J2eeServer default service was started successfully.
```

To stop the service, use the Windows `net stop` command:

```
>net stop PramatiServerdefault
The Pramati J2eeServer default service is stopping...
The Pramati J2eeServer default service was stopped successfully.
```

To remove the service permanently:

1. Run the `setup.bat` script in `<install_dir>\services\bin\` directory to set the necessary environment.
2. Run the `remove_server_default.bat` script:

```
> remove_server_defaule.bat
wrapper  | Pramati J2eeServer default removed.
```

Setting up Pramati Server as a Unix Daemon

To setup Pramati Server as a daemon on Unix based machines, execute the shell script `./pramatiserver.sh`. Ensure that you have the following files that are shipped with the distribution:

File	Description
\bin\license.txt	A text file describing the copyrights of the use of the Wrapper Service. Provided by TanukiSoftware.org.
\bin\libwrapper.so	A library file
\bin\Wrapper	This is an executable file that is invoked to install the Java application as a service
\bin\wrapper.jar	The Java Services Wrapper classes are present in this executable JAR file
\bin\realpath	
\j2eeserver\pramatiserver	A shell script that starts the server as a Daemon

File	Description
\j2eeserver\pramati_server_default.conf	A configuration file used to launch the server with pre-defined startup parameters like setting classpath, JVM options etc.
\j2eeserver\wrapper.log	A log file for the Java services wrapper. All communication between the services wrapper classes and Pramati Server are logged here.

To start Server as a daemon, run the shell script `pramatiserver` in the `<install_root>/services/j2eeserver/ directory`.

```
$./pramatiserver start
Starting @Pramati J2EE Server V 4.1@...
```

The script `pramatiserver` can be run in the following modes. The table below describes each mode:

Table 5: Server Start Modes

Mode	Description
console	When wrapper is started in console mode, the server will started as the child process of console. You will not be able to exit from the console until the Server is stopped using ***wrapper stop*** command. You will be able to see wrapper programs output on console in this mode, and hence should be used to debug setup problems
start	When **wrapper** is started in "start" mode the Server will be launched as a detached process. When running as a detached process, you will immediately regain control of the console used to launch the Wrapper. You will also be able to now close the console and the console and the Wrapper will stay running. In this mode the wrapper's output will be in wrapper.log located in bin/ dir. In both modes Pramati Server's output will be redirected to stdOut.log in logs/ dir.
dump	Irrespective of the mode server is started in you can request for a "Thread Dump" by adding the dump attributes.

The shell script internally looks for wrapper program in the `bin` directory and for `pramati_server_default.conf` in the currently directory and starts the server. To stop the server use the `stop` option.

To know more about other parameters for this script, use the `help` command:

```
$ ./pramatiserver help
Usage: ./pramatiserver { console | start | stop | restart | dump }
```

Installing Java Wrapper

To set up Pramati Server so that it will shutdown cleanly and restarts in the correct order when the system boots, create a symbolic link to the applications script within the `/etc/init.d` directory. Many applications simply copy their daemon script to this directory but Pramati Server scripts

require a symbolic link that is used to locate the rest of the application files without modifying the script.

Register proper run levels, so that Pramati Server can be used for all multi-user run levels and stopped for the halt, single-user, and reboot runlevels.

Configuring more than one Pramati Server Nodes

In the previous section we saw how to setup the Default Server as a service. We can also setup multiple nodes as services. Setting up multiple nodes as services/daemon on Windows/Unix requires editing the files in the `j2eeserver` directory. Ensure that the node you want to configure as a service is created first. A normal convention followed is to append the node name to the name of the batch scripts.

For example, to install a node, `myNode and` run it as a service, rename the files as follows:

- `install_server_MyNode.bat` for Windows. For Unix based systems it would be the `pramatiserver.sh` shell script
- `remove_server_MyNode.bat`, (only available for Windows). For Unix based systems, the `pramatiserver.sh` shell script can be used to stop the daemon service using the `stop` switch.
- `pramati_server_MyNode.conf`.

The files form a set and are required to setup/remove the services. You should have all these files separately available for different configurations of nodes hence, the convention of having the node name at the end of the filename. Edit the `pramti_server_MyNode.conf` by changing the environment variable `set.node` to the name of the node you wish to be installed as a service. In the batch scripts above edit the environment variable `_WRAPPER.CONF` to point to the correct configuration file (in this case `pramati_server_MyNode.conf`). Here is the entry in the install service batch script:

```
set _WRAPPER_CONF="%_APP_HOME%\j2eeserver\pramati_server_default.conf"
```

Note: Make sure you resolve all the conflicts that may arise due to communication ports such as the naming port, etc., before installing a new service. Also, the naming convention of appending the node name to the various files is not mandatory but recommended to avoid confusion.

Customizing Default Parameters

In the previous section we discussed setting up the server as a service. The service is started using default parameters that are picked up from the configuration file `pramat_server_default.conf` which is present in the `j2eeserver` directory. The file looks as below:

```
#*******************************************************************
# Wrapper Properties
#*******************************************************************
#EDIT following environment variables to suit local settings
set.JAVA_HOME=d:\jdk1.4.1\
```

```
set.ps_root=D:\PServer411082

#Change the node name to install another server instance as a service.
#The service will be registered with name PramatiServer<node> (PramatiServer-
default for the following setting)
set.node=default

# Java Application
wrapper.java.command=%JAVA_HOME%/bin/java
#wrapper.java.command=C:/jdk1.3.1_03/bin/java.exe

wrapper.java.mainclass=org.tanukisoftware.wrapper.WrapperSimpleApp
# Application parameters. Add parameters as needed starting from 1
# Param 1. Pramati Server Main class
wrapper.app.parameter.1=com.pramati.Server
wrapper.app.parameter.2=-node %node%
wrapper.app.parameter.3=-redirect
........
........
```

To start the service with some pre-defined parameters, the configuration file can be edited for:

- Configuring JVM Parameters
- Configuring Classpath Parameters
- Configuring Application Parameters

Configuring JVM Parameters

The following entry in the configuration file sets the JVM options. More lines can be added to suit your requirement:

```
wrapper.java.additional.1=-server
wrapper.java.additional.2=-Dinstall.root=%ps_root%/server
wrapper.java.additional.3=-Djava.security.policy=%ps_root%/server/lib/pra-
mati/pramati.java.policy
wrapper.java.additional.4=-Djava.security.auth.policy=%ps_root%/server/lib/
pramati/pramati.jaas.policy
wrapper.java.additional.5=-Dorg.xml.sax.parser=org.xml.sax.helpers.XMLRead-
erAdapter
wrapper.java.additional.6=-Dorg.xml.sax.driver=org.apache.crim-
son.parser.XMLReaderImpl
```

To add more parameters to the JVM options, append another line to this list. Note that the entries below follow a pattern, such as `wrapper.java.classpath.<n>` where `n` is incriminated starting

from value 1. Make sure that the number parameters are in serial order without any missing numbers.

To set the JVM Heap Size values, set values for the following parameters in the configuration files:

```
# Initial Java Heap Size (in MB) equivalent to -Xms
wrapper.java.initmemory=16
# Maximum Java Heap Size (in MB) equivalent to -Xmx
wrapper.java.maxmemory=64
```

Configuring Classpath Parameters

You can add additional JAR files or directories containing class files to the Server classpath. To add a JAR file, append the list below with a similar entry. Note that the entries below follow a pattern as explained in the previous section. Ensure that the number parameters are in serial order without any missing numbers. Wild card characters such as "*", "?" can also be used.

```
# Java Classpath (include wrapper.jar)  Add class path elements as needed
starting from 1
wrapper.java.classpath.1=%ps_root%/services/bin/wrapper.jar
wrapper.java.classpath.2=%ps_root%/server/lib/pramati/server_patch.jar
wrapper.java.classpath.3=%ps_root%/server/lib/pramati/classpath.jar
wrapper.java.classpath.4=%JAVA_HOME%/lib/tools.jar
wrapper.java.classpath.5=%JAVA_HOME%/jre/lib/rt.jar
```

Troubleshooting Tips

If anything goes wrong while installing the service, use the logs in the directory of the node which is configured. The `Wrapper.log` tells us if anything has gone wrong while installing the service. Set the debug environment variable to true at the end of the configuration file `pramati_server_<node_name>.conf`. The entry in the file is:

```
#wrapper.debug=TRUE
```

Pramati Server Components

Pramati Server is made up of the following configurable components:

- Pramati Server Standalone node
- Pramati Server Cluster EJB node
- Pramati Server Cluster Web node
- Pramati Server Cluster BOTH node (refers to a node that is both EJB and Web)
- Pramati Server Cluster node
- Pramati Server embedded Message Server
- Pramati Server Standalone Message Server node
- Pramati Server Load Balancer
- Pramati Server Administration Service
- Console
- Java clients
- Third Party integration with Apache /IIS using WebGate
- Integration with Oracle Forms

To configure the above components, Server provides the following means:

- Pramati Server Administration Service
- Pramati Server Management Console
- Modifiable Configuration XML files

Configurable Server Components

The following sections explain the configurable Server components mentioned above.

Pramati Server Standalone node

Pramati Server is a Java implementation of the Java™ 2 Platform, Enterprise Edition standard. It is developed to simplify the process of deploying EJBs and other J2EE components.

Pramati Server Cluster node

A Cluster is a group of server nodes that work together to provide a more powerful and reliable application platform. The Cluster is transparent to the user and provides the

abstraction of a single processing server for operations such as configuration, tuning, deployment and monitoring.

A Pramati Server Cluster node can be of three types: EJB, Web and BOTH. Pramati Server can be configured as a Standalone node or a Cluster node. Application components may be deployed on the Server in one of the following ways:

- Locally (without Load Balancer): The application is deployed individually on one or more independent Standalone nodes on the network. An external or Pramati HTTP Server directs requests to each Server
- Clustered (with Load Balancer): Multiple Cluster nodes act as a single Server and balance application request loads.

Pramati Server Cluster EJB node

In an EJB node, only the EJB container gets started and the Web container does not. This is advantageous when the application consists of only EJBs.

Pramati Server Cluster Web node

A Web node is used when the application uses static HTML, JSP and Servlets. A Web node configured as a Standalone server can act as a Load Balancer.

Pramati Server Cluster BOTH node

While configuring a Cluster node, the Server Administration Service provides the option to select EJB or Web or BOTH type of node. The BOTH option is selected when the application uses Java, EJBs, JSP and Servlet components. This options can be enabled while configuring a new node.

Pramati Embedded Message Server

Embedded Message Server runs in the same VM as Pramati Server. It is part of the Default Pramati Server configuration and can be optionally started every time the Server starts. This can not be a part of the EJB/WEB/BOTH node.

Pramati Server Standalone Message Server node

Pramati Message Server can be set up as an independent, full-function Pramati Message Server to process enterprise Java messaging applications. The Pramati Message Server High Availability (HA) Solution enables failover for JMS applications.

Pramati Server Load Balancer

Pramati Server Load Balancer is an HTTP-aware Web Traffic Management Interceptor that acts as a request dispatcher and performs intelligent forwarding of requests based on pattern, load, session etc. It integrates with other commercial application servers, thus making it a cost effective solution for many enterprises willing to adapt a sophisticated traffic management setup.

Pramati Server Administration Service

Pramati Administration Service can be secured through a username and password. Two services can communicate with each other only if both use identical username and password.

Pramati Server Management Console

The Console is customizable to suit server administration requirements. Through diagnostic and tuning aids, administrators can optimally deploy resources and extract maximum performance in any scenario.

Java Client

These are Java programs that directly access EJB applications or other services over RMI.

Third Party Integration with Apache/IIS

Pramati provides WebGate plugin that enables Pramati Server to comminucate with other web servers like Apache and IIS. This is useful in scenarios where a web server can serve all static content and Pramati can take care of serving the dynamic pages. This way, Pramati server is relieved of the burden to server static pages.

To learn more about configuring Pramati Server with this support, refer to the *'Administration Guide'*.

Configuring Components using Pramati Server Administration Service

Pramati Administration Service enables discovery, configuring, and managing Pramati Servers in the network. This service installs with every Pramati Server installation. For management of a Server, the service must be started and should run on that Server host.

When a service starts, it discovers all other Administration Services that are up on the network at that time. It then lets you connect to those Servers through the Console and enables remote management of that environment.

Configuring Components using Console

Console uses the Java Management Extensions (JMX) framework and is a web-based tool to manage one or more Pramati Servers.

Configuring Components by Modifying Configuration XMLs

Pramati Server instances can be configured manually by modifying the configuration XML files. Server instances of type J2EE, Message Server and Cluster can be manually created with simple configuration in the XML files.

For details on using XMLs to configure the various components, read the chapter '*Creating and Configuring Pramati Server Instances Using XMLs'.*

Where and how to configure each component

Table 6: List of components of Pramati Server that are installed and can be configured

Component	Where	Install	Configure
Pramati Server Administration Service	Server machine	Automatically installed as part of Server installation (PServerXX.exe or .jar)	Needs one-time setup of the service/ daemon using install_services.bat (on unix, prs) scripts under the /services directory.
Pramati Server Standalone node	Server machine	Install PServerXX.exe (or .jar)	Configured using the Console as a StandAlone node. In a single Pramati Server installation, multiple instances can be configured, each identified by a unique name and port.
Pramati Server Cluster	Server or Cluster node machine	Install PServerXX.exe (or .jar)	Configured using the Console as a Cluster of Nodes. In a single Pramati Server installation, multiple Pramati Server Clusters can be configured.
Pramati Server Cluster EJB node	Server or Cluster node machine	Install PServerXX.exe (or .jar)	Configured using the Console as a Cluster EJB node.
Pramati Server Cluster Web node	Server or Cluster node machine	Install PServerXX.exe (or .jar)	Configured using Pramati Server Console as a cluster node. In a single Pramati Server installation multiple nodes can be configured with a unique name and can be started along with the server at the same time.
Pramati Server Cluster BOTH node	Server or Cluster node machine	Install PServerXX.exe (or .jar)	Configured using the Console as a Cluster BOTH node. From a single Pramati Server installation, multiple nodes can be configured, having unique names and can be started synchronously with the Server.
Java Clients	Client workstations	No separate installation. From the installation on the Server machine, copy the /lib directory to the client machine.	Modify jndi.props to look up objects on the Server.
Pramati embedded Message Server	Server machine	Installs with every Server (PServerXX.exe or .jar)	The embedded Message Server starts by default. Option is available from the Console to disable JMS in Server Configuration.

Table 6: List of components of Pramati Server that are installed and can be configured

Component	Where	Install	Configure
Pramati Server Standalone JMS node (or Pramati Message Server)	Server machine	Installable available only on product CD. Run PMsgServerXX.exe or .jar.	Configured using the Console.
Pramati High Availability Message Server	Server machine	Installable available only on product CD. Run PMsgServerXX.exe or .jar.	Configured from the Console.
Third party integration with Apache/IIS using WebGate	The WebGate plug-in should be installed in the external server to be used.	Plug-in available for downloads from http://www.pramati.com/	Configured using the plug-in and editing configuration files in the install directory of the web server to be integrated. Installation instructions provided in the Administration guide. Separate instructions are specified for both Apache and IIS in respective chapters.
Web Enabling Oracle Forms Applications	server machine.	Requires configurations in the server.	

Adding a User and Changing Password

After installing the Pramati Server, you may want to add users and change the default password. For information on how to add and configure Groups and Realms, read the chapter 'Configuring Security for Pramati Server' in the Pramati Server 4.1 Administration Guide.

Adding User

To add a user using the Console:

1. Start and connect to Server.
2. Connect to the Administration console using a browser by typing the URL http://<host-name>:<port-number>/admin/. For example, if the server is installed locally with default HTTP port, the URL would be http://localhost:8181/admin/. This brings up the web-based Administration Console login screen. The default username and password are root and pramati respectively.
3. On the Console, select Configure > Security in the Explore panel.
4. Click 'Settings' for an existing realm.
5. Under the Groups section, click 'Add User'. This leads you to the 'Security > Realm > User' screen for adding a user. Provide the following details for the new user:

Table 7: Adding a User

Fields	Description
Name	Name of the new User
Password	To authenticate the new user name
Retype Password	Confirm your password here
Groups	Select the available Groups to which the new User is to be added using the '>>' option

Click 'Save'. This adds the user to the selected group, that shows an increment in its number of Users field. Clicking the Group name link displays the details for the User.

If no Group has been selected while adding the User, he is added to the 'Everybody' group, and can be later on assigned to any group.

To make changes to the password and the group fields, click edit against the selected user.

Deleting User

A user can be deleted specifically from a group, or can be completely removed from the Server.

To delete a user from a specific group:

1 Click the required group. The list of existing users and permissions appears in the Group page.

2 Select the user to be deleted from the list using the checkbox, and click 'Delete'.

This deletes the selected user from just the selected group.

To remove a user from the server:

1 Select Configure > Security > Edit > All Users. The list of existing users and permissions appears.

2 Select the user to be deleted from the list using the checkbox, and click 'Delete'.

Changing Password

To change the password for the Console:

1 Select Start > Programs > Pramati 4.1 > Server Administration.

2 Enter the user name as '*root*' and password as '*pramati*' and press <enter> to login to the Console.

3 On the Console screen, select Configure. There are two ways for changing the password:
- Using the User Settings
- Using the Security option

Using User Settings

Select Configure > User Settings in the Explore panel. This takes you directly to a screen where you can modify the password settings.

Enter the old and the desired password. Confirm the password and click 'Save'. The password is changed and should be used the next time you log in.

Using Security option

Select Configure > Security in the Explore panel. Click 'Settings' against the realm name, the password for which has to be changed. This leads you to the 'Security > Realm' screen that displays details of the selected realm:

- Groups
- Permissions
- Login Module
- User Manager

Changing password for User

Under the 'Groups' section, click 'All Users'. This takes you to the 'Security > Realm > Users' screen. Click 'Settings' for the user for which the password is to be changed. This takes you to a screen for editing the following details:

Table 8: Changing Password for the selected Group

Fields	Description
Old Password	The existing password for the user
New Password	The new password for the user
Retype New Password	Retype the new password for the user
Group List	The groups to which the user belongs

The name of the user is displayed by default. Click 'Save' to go back to the 'Security > Realm' page.

Introduction to Pramati Server Administration Service

Pramati Server is configured and started using the Server Administration Service. The service equips you to:

- Discover various Server instances running on the sub-net through a discovery mechanism based on User Datagram Protocol (UDP)
- Configure and start:
  - Standalone Server
  - J2EE Server Cluster
  - Message Server
  - Message Server Cluster
  - Load Balancer

Role of Administration Service

The Server Administration Service (Admin Service) enables you to configure, start and connect to any Server in the LAN. This is a Standalone Java program that can be configured as a Windows NT Service or a Unix daemon and is available at system startup. You can operate the Server through a browser or from the command line using Admin Service as a gateway.

This service is required to start and connect to a:

- Cluster node configured on a remote host
- Standalone server instance

Admin Service should be started on every computer on which you want to configure, start, or manage servers. This facilitates remote accessibility and communication with the Remote Administration Services (RAS).

You should provide the correct username and password for using the Service. Only those administration services that have the same username and password can communicate with each other. The Service starts by discovering Pramati Servers that are running on the LAN. These Servers are displayed in the browser along with their respective URLs.

Admin Service is shipped and installed as a standard server component in the same installation directory as the Server. A set of batch files and shell scripts is provided to install the startup service separately on Windows NT and Unix platforms.

Note: *The Service runs on port 4748. To initialize the Service, it is essential to keep the port open at all times.*

How does Server Administration Service work

The Admin Service keeps track of other services running on the LAN using a discovery mechanism, that enables you to manage server configuration from a single administrator host machine.

Starting Server Administration Service

To start the Server Administration Service:

1. Choose Start > Programs > Pramati 4.1 > Server Administration. This displays a screen with options for either configuring and starting a cluster, or connecting to a running server. If you select the second option, provide the details for the IP address and the HTTP Port where the Server Administration Service is running. If you provide the details for where the Console is running, you will be taken to the Console Home page.
2. Selecting the option for configuring and starting a cluster, and clicking OK displays the login screen for Server Administration Service. Provide the user name as 'root' and password as 'pramati' and click 'Login'. This displays the Server Administration Service screen.

Configuring Standalone Pramati Server Using Admin Service

11

This section talks about starting the Default Server on Unix and Windows, as also configuring a Standalone Pramati Server using the Server Administration Service.

Using Default Server

When you install the Server, a default configuration is created that appears in the list of managed Servers as `default`. The Default Server starts if you do not specify another server. You can specifically start the Default Server without starting the Server Administration. The standard configurations for the Default Server are:

- Server name: `default`
- Host: `Localhost`
- Lookup Port: `9191`
- HTTP Port: `8181`
- User name: `root`
- Password: `pramati`
- Realm: `system`

Starting the Default Server on Windows

When you install Pramati Server, a Default Server is automatically configured. To start it without launching Server Administration Service:

1. Choose Start > Programs > Pramati 4.1 > Default Server
2. To login, enter username as *root* and password as *pramati*
3. Click 'Login'

The Default Server starts and the Console Home page appears.

Starting the Default Server on Unix

To start the default Server in Unix, run `runDefaultServer.sh` located at <install_dir>/server/bin. To login, enter username as *root* and password as *pramati*.

Configuring Standalone Server

Managing a Standalone Server involves creating, modifying and removing a Standalone Server using the Server Administration Service.

Creating Standalone Server

To create a Standalone Server using the Server Administration Service:

1. Start the Server Administration Service. Read 'Using Pramati Server Administration Service' for details.
2. Click Standalone Server in the Explore panel. The 'Add Standalone Server' page appears in the Display panel.
3. Enter the following details:

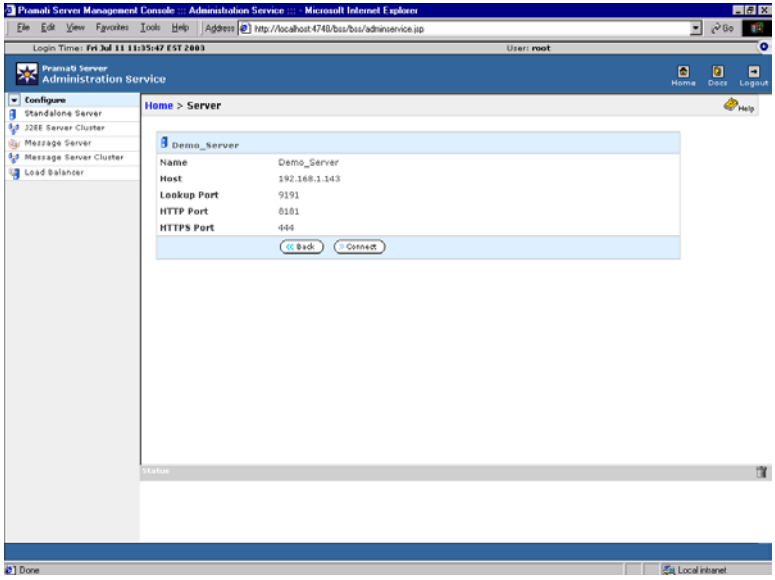
Table 9: Details needed for configuring a Standalone Server

Fields	Description
Name	Name for the new server, say, Demo_Server. This is a mandatory field.
Host	The Host machine name (IP) on which the Server is to be configured, say, localhost. This can be any machine in the LAN where the Server Administration service is running. This is a mandatory field.
Lookup Port	The Lookup Port on the host machine on which the Server will start, say 9192. This is a mandatory field.
HTTP Port	The HTTP Port on which the Web container is started, say, 8182. This is a mandatory field.
HTTPS Port	The HTTPS Port on which the Web container is started, say, 546. This optional field is used for secure communication.

A new server instance 'Demo_Server' running on localhost is added. This takes you to the Server Administration Service main page where the added server appears in the list of managed Standalone servers.

Viewing details of added Standalone Server

To view the details of the added server, click its name that is provided as a link on the Home page. This takes you to the 'Home > Server' screen showing the following details:



Configuring Standalone Server

To configure (verify or modify) the details for the added server, click on the configure button provided. This takes you to the 'Home > Standalone Node' screen where you can modify details for Lookup Port, HTTP Port, HTTPS Port and Enable HTTPS fields. Save the modifications made. The Lookup Port and HTTP Port fields cannot be left blank.

The Standalone Server can be started from this page itself using the 'Start' button.

Note: *You can modify Server properties only when the Server is not running. For further details on configuring a Server using the Command Line, read 'Managing Pramati Server Using Command Line'.*

Starting Standalone Server

To start the Server:

1. You can either click 'Start'
   - against the required server on the Server Administration Service Home page, or
   - on the 'Home > Standalone Node' screen used for configuring the Standalone Node.
2. The 'Home > Server', 'Start Standalone Node '<name_of_server>' screen appears.
3. The fields Host, Lookup Port, and HTTP Port, on which the Web Container was started, cannot be edited. These values are the same that were used while adding the server, and are displayed as default.

4 Provide the following details:

Table 10: Details needed for starting a Standalone Server

Fields	Description
Realm	Provide here the Realm name. By default, this appears as *system*. This is a mandatory field.
Username	Provide here the Username for the specified Realm. By default, this appears as *root* for the *system* Realm. This is a mandatory field.
Password	Provide here the password for authenticating the Username. For the Username *root*, the default password is *pramati*. This is a mandatory field.
JVM options	Provide here the JVM options, (-D or -X) if you want to pass any Java options.

The Status panel displays the related messages as the Standalone Server is started.

Connecting to Standalone Server

Once the Server starts, the 'Start' button is replaced by the 'Connect' button. To connect to the Server:

1 Click 'Connect'. This displays the login screen for the Console.

2 Enter the relevant user name and password. For the user name *root,* the password is *pramati*. Click 'Login'.

3 The Console Home page appears. The configuration details of the connected server are displayed in the 'Server Configuration' section. The name of the new server is displayed in the Details panel on the Console.

Stopping Standalone Server

To stop a connected server, click the 'Stop Server' link on the Console page. This prompts you to confirm if you really want to stop the server.

Clicking 'OK' stops the running server, and displays the message for the same. On the Server Administration Service Home page, click 'Refresh' button provided. The 'Connect' button for the new server changes back to 'Start'.

Note: *You must refresh the main page to verify the status.*

Removing Standalone Server

You can remove a server only if it is not running. To remove a server:

1 Stop the Server using the Console.

2 On the Server Administration Service Home page, scroll down and click 'Refresh'. This displays the 'Connect' button for the Standalone server (Demo_Server here) as changed back to 'Start'.

3 Select 'Demo_Server' using the check box provided.

4 Click 'Delete' to remove the server.

Adding Web Load Balancer using Administration Service

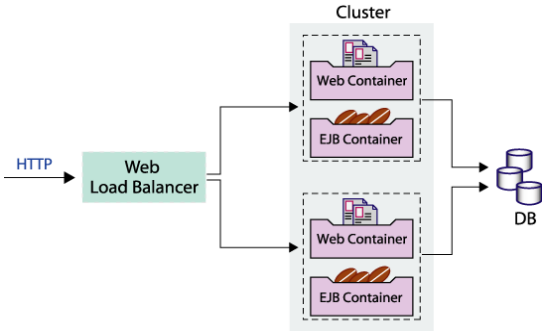
Pramati Server provides a software based load balancing solution. It is implemented as an Interceptor which is a part of the Interceptor stack in the Pramati Web Container. Configuring the Load Balancer is as simple as adding entries into the `web-lbconfig.xml` file. Pramati Server's Load Balancer intelligently distributes load among a group of nodes. While distributing load it also takes into consideration factors like session stickiness, thus sending session based requests to the node where the session was initiated. It also acts like a normal Load Balancer, a URI based director and also a failover node.

This chapter describes the steps to create, modify and remove a Pramati Load Balancer using Server Administration Service. Before getting into the details of setting up the load balancing, let's discuss a few scenarios.

To know more about the concepts and architecture of the Load Balancer, please refer to the chapter "Request Dispatcher Architecture", in the Technical Reference guide.

Load Balancing Scenarios

In this section, we'll discuss a number of scenarios where the Request Dispatcher in Load Balancer can be used. A typical load balancing scenario is depicted by the following diagram:

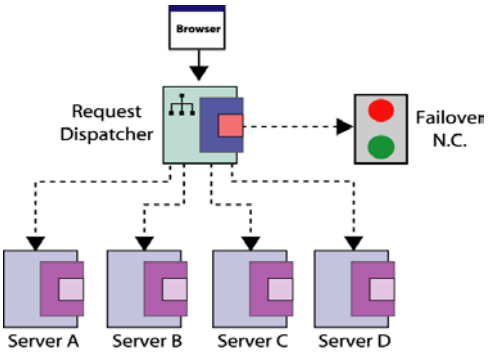


Case 1 The Request Dispatcher (RD) intercepts all requests to a set of homogeneous machines. In this case the dispatcher would not serve any request on its own:

Typical Setup:

- RD as a Load Balancer

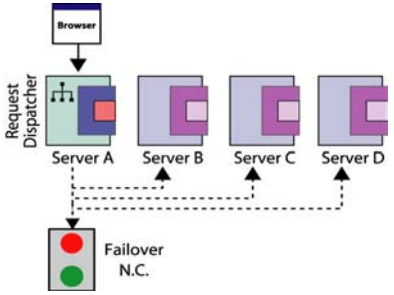
- RD for High Availability
- RD for Load Balancer and High Availability



Case 2 The node which runs the Request Dispatcher, would also be serving some request.

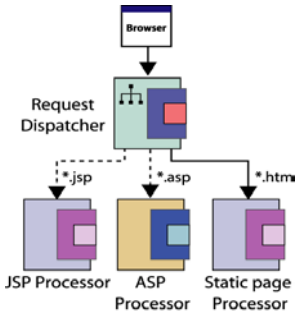
Typical Setup:

- The request dispatcher allows plug-ins to be written for Pramati Server
- RD with default URI/Failover/Session stickiness plug-in, which allows forwarding requests based upon URI:



Case 3 A pure Request Dispatcher for various machines based upon URI. For instance all JSP requests will be forwarded to one node, all ASP requests would be forwarded to one node, and all static file requests will be forwarded to another node.

- RD in the Demilitarized Zone and all the nodes beyond the inner firewall.



Case 4 Request Dispatcher working along with a Pramati Plug-in for Apache/IIS to enable distribution of the load to backend Pramati Servers.

Pramati Server WebGate communicating with the Request Dispatcher, which distributes load and provides failover facility.

Adding a Load Balancer node

To add a Load Balancer node:

1. Start the Server Administration Service.
2. Click 'Load Balancer' in the Explore panel. The 'Add Load Balancer Node' page appears in the Display panel.

3 Enter the following details:

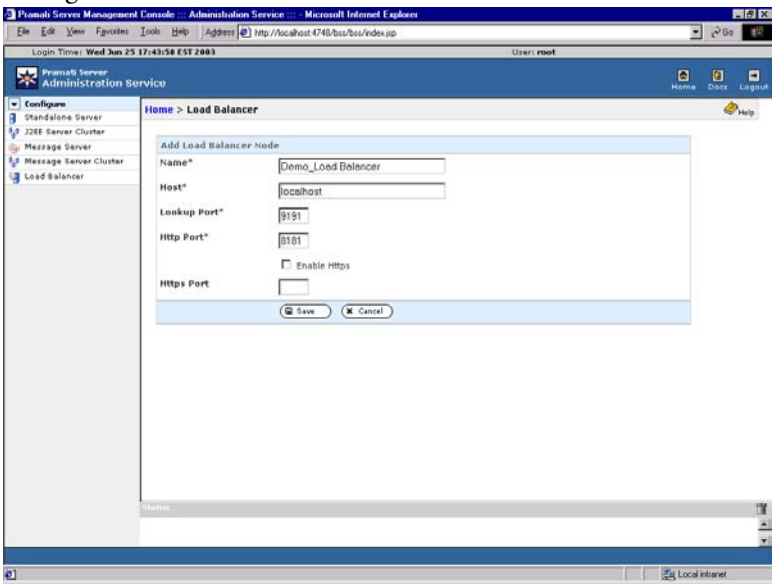


Table 11: Details needed for configuring a Load Balancer

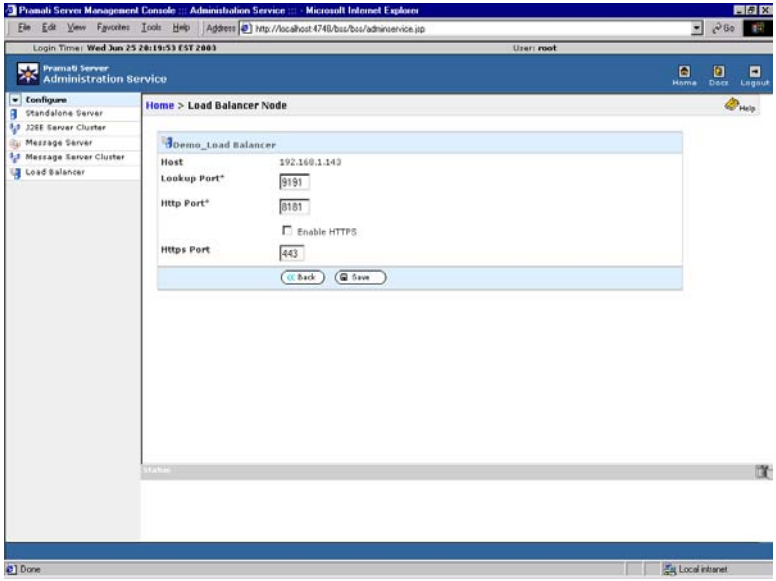
Fields	Description
Name	Name for the Load Balancer, say, Demo_Load Balancer.
Host	The Host machine name (IP) on which the Load Balancer is to be configured, say, 192.168.1.143. This can be any machine in the LAN where the Server Administration service is running.
Lookup Port	The Lookup Port on the host machine on which the Load Balancer will start, say 9191.
HTTP Port	The HTTP Port on which the Web container is started, say, 8181.
HTTPS Port	The HTTPS Port on which the Web container is started, say, 444. This is used for secure communication.

Optionally enable HTTPS for secure communication, and click ‘Save’.

A new server instance ‘Demo_Load Balancer’ running on localhost is added with its naming service listening on Port 9191 and HTTP server on Port 8181. This takes you back to the Administration Service main page where the new server appears in the list of managed servers.

Configuring Load Balancer

To configure (verify or modify) the details for the Load Balancer, click ‘Configure’. This takes you to the ‘Home > Load Balancer Node’ screen.



You can modify details for lookup port, HTTP Port, HTTPS Port and Enable HTTPS fields. Save the modifications made.

Viewing Load Balancer details

To view the details for the Load Balancer, click its name which is provided as a link on the Service page. This takes you to the 'Home > Load Balancer' screen.

Starting Load Balancer

To start the Server:

1. On the Server Administration Service main page, click 'Start' against the required server.
2. The 'Main Page > Start StandAlone Server' page appears.
3. The fields Name, Host, Lookup Port, and HTTP Port, on which the web container was started, cannot be edited. These values are the same that were used while adding the server, and are displayed as default.
4. You need to provide the following details:

Table 12: Details needed for starting a Server

Fields	Description
Realm	Provide here the realm name. By default, this appears as *'system'.*
Username	Provide here the user name for the specified realm. By default, this appears as *'root'* for the *'system'* realm.

Table 12: Details needed for starting a Server

Fields	Description
Password	Provide here the password for authenticating the user name. For the user name *'root',* the default password is *'pramati.'*
JVM options	Provide here the JVM options, if you want to pass any Java options.

Connecting to Load Balancer

To connect to the Server:

1. Once you have started the Server, the started Server is shown as a blue icon, and 'Start' is replaced by a 'Connect' button.
2. Clicking 'Connect' takes you to the login screen for the Console.
3. Enter the user name as '*root'* and password as '*pramati'*. Click 'Login'.
4. The Console Home page appears. You can see the configuration details of the connected Server in the 'Server Configuration' section.

Stopping Load Balancer

To stop a connected server, click 'Stop' in the Console. To exit from the Console, click 'Logout'.

In the Administration Service main page, the status appears as "stopped".

Note: *You must refresh the main page to verify the status.*

Removing a Load Balancer

You can remove a Load Balancer only if it is not running. To remove a Load Balancer:

1. On the Server Administration Service main page, select the Load Balancer that is to be removed.
2. Click 'Delete'. This deletes the selected Load Balancer.

Configuring Web Load Balancer Using XML

13

A Load Balancer works as a web-request dispatcher, distributing the web requests across a set of Web/J2EE Cluster nodes. In advanced configurations, it distributes the web requests even across non-cluster server instances and multiple vendor's servers. Load Balancing for EJB clusters is done at the container level and does not need any external configuration.

A Load Balancer is a variant of Web node. The steps involved in setting up a Load Balancer are:

1. The DIR and the config files have to be setup just as one would do for a Standalone instance.
2. Copy the `server-config.xml` and `web-lbconfig.xml` into the config DIR
3. Enable the Load Balancer in `web-config.xml`

```
<request-dispatcher enabled="true" name="web-lbconfig.xml"/>
```

4. Configure the Load Balancer properties in `web-lbconfig.xml`. The nodes are referenced by their name:

```
    <node name="first" type="pramati" host="localhost" web-
port="8484">
      <naming-port>9494</naming-port>
      <socket-pool min="20" max="50" idle-time-out="1000"/>
    </node>
```

You can also configure node chooser and node group information from `web-lbconfig.xml`. For more information on node chooser, node group and failover refer to the *Technical Reference* document.

The following table lists all the parameters for configuring a Web Load Balancer:

Table 13: Dispatcher Configuration Parameters

Tag Attribute	Option	Description
node		
	name	A unique name specifying the node
	type	Type of cluster
	host	Host name
	web port	Port where the node listens for requests
	socket pool (min)	Minimum socket pool size
	socket pool (max)	Maximum socket pool size

Table 13: Dispatcher Configuration Parameters

Tag Attribute	Option	Description
	ping interval	Number of seconds after which the other nodes are sent a request
	Idle time out	Request time out in seconds
node group		
	name	A unique name specifying the node group
	type	Type of cluster
	node-ref	The name by which the node is referred
node chooser		
	name	A unique name specifying the node chooser
	class	The implementation class for this node chooser
	host name	Node chooser host name
	URI Pattern	URI pattern for mapping requests
failover node chooser		
	name	Name specifying the failover node chooser
	class	The implementation class for this node chooser
	type	Type of cluster
	enable	State of the node chooser

Configuring J2EE Cluster Using Server Administration Service

14

All cluster configuration details are maintained in a database as well as in the configuration files. Pramati Server provides the following ways to configure J2EE Clusters:

Using Server Administration Service

- Using Command Line (read the chapter)
- Modifying XMLs

Configuring J2EE Cluster

To configure a J2EE Server Cluster using Server Administration Service:

1. Start the Server Administration Service.
2. Click the J2EE Server Cluster option in the Explore panel. This displays 'Step 1 of 3: Add Cluster' in the Display panel. Provide details for the following fields:

Table 14: Step 1 of adding a J2EE Cluster

Fields	Options	Description
Name		Specify the Name of the new cluster. Say, 'Test'. This is a mandatory field.
EJB Cluster		The EJB Cluster is shown selected by default.
Web Cluster		Select whether you want this to be a Web cluster using the 'Yes' and 'No' options provided.
Database		The database stores the cluster configuration details. In this section, specify the following mandatory details:
	URL	Specify here the URL of the database. Say, 'jdbc:oracle:thin:@db1.pramati.com:1521:db1'. This is a mandatory field.
	Driver	Specify the database driver name. This driver interfaces between the J2EE server and the database. Say, 'oracle.jdbc.driver.OracleDriver'. This is a mandatory field. The driver jars are not picked up from a common location, but are taken from the path specified in the input box. In case the user does not specify the path, the server attempts to pick them up from the /lib/ext directory under the server installation directory, or from the system classpath.
	Username	Specify the username used to connect to the database. For example, for an Oracle database, the username can be 'scott'. This is a mandatory field.
	Password	Specify the associated password used to connect to the database. For example, for an Oracle database, the password can be 'tiger'. This is a mandatory field.

Table 14: Step 1 of adding a J2EE Cluster

Fields	Options	Description
	Table	Provide a table name for the database, say 'emp'. This is a mandatory field.

Click Next>>. 'Step 2 of 3: Add Nodes' page appears.

Table 15: Step 2 of adding a J2EE Cluster

Fields	Description
EJB, Web	Specify the type of the node. This can be EJB or Web. You can also select both the types.
Name	Specify the name of the node. Say, 'MyNode'.
Host name	Specify the host name or IP address of the machine on which the node will run. Say, 'localhost'.
Lookup port	Specify the Lookup port on which Naming service should start. Say, '9191'.
HTTP port	Specify the HTTP port on which the Web container will start. This is valid only when you specify the node type as Web or Both. Say,'8181'.
HTTPS port	Specify the secure communication port (HTTPS) on which you want the Web container to start. This is valid only when you specify the node type as Web or BOTH. You can enable or disable this port.

To add more than one node, provide similar details in the fields provided.

Enter the Session Persistence details. Select whether EJB and Web will have session persistence 'In Memory' or 'Database'. This makes sure that the state of a stateful session bean is passivated into a database, and that the state of the bean is replicated on any running node during failover. This database can be different from the database that stores the cluster details. Click 'Next'.

A successful node addition prompts 'New Cluster Test created successfully.' in the Status panel. The third step is to set the Session properties in case a failover happens. In the new page, you can:

- Replicate data on all the nodes of the cluster, or
- Maintain a backup on selected nodes of the cluster

Click 'Save' to return to the Server Administration Service main page. The new cluster is seen listed here.

Modifying Cluster

On the Server Administration Service, click 'Configure' against the cluster that is to be modified under J2EE Server Clusters section. A screen displaying the cluster details and the nodes that define the cluster appears.

You can:

- View configuration details of individual nodes by clicking on their names provided as links.
- Specify the weight of the cluster nodes. This value is used for Load Balancing.

- Configure nodes for Lookup, HTTP and HTTPS Port values.
- Add new nodes to the cluster by providing values as Name, Type, Host, Lookup, HTTP and HTTPS Port values.
- Delete existing nodes.
- Edit the failover details for the Cluster.

Click 'Save'.

Starting Cluster

To start the cluster:

1. On the Server Administration Service, click 'Start' against the required cluster under J2EE Server Clusters section. This displays details of the nodes that are defined for the cluster along with an option to specify JVM options, and the Login details of the user.
2. Select all the nodes that you want to start.
   Note: *You can click on the name of the node to view the node details.*
3. Enter the password for the relevant realm. The default value for the '*system*' realm is *pramati*.
4. Click 'Start'.

The Status panel displays the related messages as the specified cluster nodes start. A successful start displays the option to 'Connect' to the cluster.

Connecting to Cluster

To connect to the cluster:

1. Click 'Connect' against the cluster. The login page appears.
2. Enter the username and password for the realm. For the system realm, the default values are *root* and *pramati*.
3. Click 'Login'.

You are connected to the selected node and details related to the connected node are displayed on the Console Home page.

Deploying an application on Cluster node

Deploying an application on a cluster node is the same as deploying it on a Standalone Server. However:

- An application deployed on one node is available to all the running nodes of the cluster.
- Even if a cluster node (say N2) is shutdown when the application is deployed on another cluster node (say N1), the application is available on N2 after you start it.
- All resources used by an application, such as Datasources, Connector, Mail, etc. are available on all the nodes of a cluster.

To deploy an application on a cluster node:

1. Start and connect to the cluster node.
2. Click Start > Programs > Pramati 4.1 > Deploy Tool. The 'Connect To Server' window appears, and asks you to enter the password for the realm '*system*' and user name '*root*'. The default password to be entered is '*pramati*'. Click 'Connect'.
3. The Deploy Tool UI appears.
4. Click Archive > Open. Browse to the location of the application that is to be deployed.
5. The number of tasks that you need to resolve appear in the Status bar. Resolve all the tasks, and click Archive > Deploy.
6. The application is automatically deployed on all the running nodes of the cluster.

Effects of deploying an application on Cluster node

Deploying an application on a cluster node is the same as deploying it on a Standalone Server. However, these additional factors are valid for a cluster:

- Any application deployed on one node of the cluster is automatically deployed on all other running nodes in that cluster.
- Automatic application synch-up occurs when a node starts. If a cluster node (say N2) is not running when the application is deployed on another cluster node (say N1), the application is available on N2 after you start it.
- Any change in the application state is reflected on all the nodes.
- Application resources and security realms are replicated across the nodes.

Location of the extracted application

An application deployed on a specific cluster node is extracted to:

```
<install_dir>/server/nodes/<node_name>/archives/
```

Stopping Cluster

To stop a Cluster, use the 'Stop Cluster' option provided on the Console. This prompts you for a confirmation for the action. Selecting 'OK' stops the Cluster, and you are notified.

You need to explicitly click on the 'Refresh' button provided on the Server Administration Service page to display the current status of the cluster. Once the cluster is stopped, a checkbox is made available for deletion of the same on the Server Administration Service.

To stop a connected node of a Cluster, use the 'Stop' option provided after selecting the node on the Console.

Deleting Cluster

You can delete a cluster only when it is stopped.

To delete a cluster, select it using the checkbox provided, and click 'Delete'.

Configuring Message Server as a Standalone Node

The Standalone Message Server can be installed and configured separately from the J2EE Server using the Server Administration Service.

Configuring Standalone Message Server

This section describes the steps to create, modify and remove a Standalone Message Server using the Server Administration Service.

Creating Standalone Message Server

To create a Standalone Message Server on the Server Administration Service screen:

1. Start the Server Administration Service.
2. Click 'Message Server' in the Explore panel. The 'Add Standalone Message Server' page appears in the Display panel.
3. Enter the following details:

Table 16: Details needed for configuring a Standalone Message Server

Fields	Description
Name	Name for the new standalone message server, say, 'Demo_MsgServer'.
Host	The Host machine name (IP) on which the standalone message Server is to be configured, say, '192.168.1.143'. This can be any machine in the LAN where the Server Administration service is running.
Lookup Port	The Lookup Port on the host machine on which the Server will start, say '9191'.
Fields	**Description**
Name	Name for the new standalone message server, say, 'Demo_MsgServer'.

4. Click 'Save'.

A new server instance 'Demo_MsgServer' running on the host 192.168.1.143 is added, with its naming service listening on Port 9191. This takes you back to the Server Administration Service Home page where the new server appears in the list of managed servers.

Configuring Standalone Message Server

To configure (verify or modify) the details for the Standalone Message Server, click the 'Configure' button. This takes you to the 'Home > Message Server' screen.

You can modify details for the Lookup Port (mandatory field), while the Host address is shown as a non-editable field. Save the modifications made. You can even start the specific Server from this page using the 'Start' button.

Starting Standalone Message Server

To start the Server:

1. On the Server Administration Service Home page, click 'Start' against the required Standalone Message Server.
2. The 'Home > Message Server' page appears displaying the following fields to be specified to start the 'Demo_MsgServer'.
3. The fields Name, Host, and Lookup Port cannot be edited. These values are the same that were used while adding the server, and are displayed as default. You need to provide the following details:

Table 17: Details needed for starting a Standalone Message Server

Fields	Description
Realm	Provide here the Realm name. By default, this appears as *'system'*.
User Name	Provide here the user name for the specified Realm. By default, this appears as *root* for the *'system'* Realm.
Password	Provide here the password for authenticating the user name. For the user name *root*, the default password is *'pramati'*.
JVM options	Provide here the JVM options, if you want to pass any Java options.

Viewing details of Standalone Message Server

To view the details for the new server, click its name that is provided as a link on the Server Administration Service page. This takes you to the 'Home > Message Server' screen showing the details for the Host and Lookup Port.

Once the server has started, you can connect to it from this screen using the Connect button provided.

Connecting to Standalone Message Server

To connect to the Standalone Message Server:

1. Once you have started the Standalone Message Server, click 'Connect'. This takes you to the login screen for Console.
2. Enter the user name as '*root*' and password as '*pramati*'. Click 'Login'.
3. On successful login, the Console Home page appears. You can see the configuration details of the connected server in the 'Server Configuration' section.

Stopping Standalone Message Server

To stop a connected Standalone Message Server, click 'Stop Server' on the Console. To exit from the Console, click 'Logout'.

In the Server Administration Service Home page, the 'Connect' option changes back to 'Start'.

Note: *You must refresh the main page to verify the status.*

Removing Standalone Message Server

You can remove a Standalone Message Server only if it is not running. To remove a server:

1. On the Server Administration Service Home page, select the server that is to be removed.
2. Click 'Delete'.

Using Loaded Send Queues

Loaded Send operation in Pramati Message is used in situations where there is a fast sender and a slow receiver. Normally, messages are kept in-memory. If the number of messages that are sent are increased, this may increase the memory usage infinitely. There are two ways to overcome this limitation:

- Setting Max Queue Size based on the application activity
- Loaded Send

Setting the max queue size to the maximum possible value based on the activity of the application however has its disadvantages. Here the client cannot send messages beyond the maximum size specified, in case messages cross the max limit.

To overcome this, the loaded send queues can be used. In the next section we will discuss how to configure the Message Server for loaded queues.

Configuring the Message Server for Loaded Queues

The following components are required to configure the Message Server for loaded queues.

- A properties file
- A temporary store for storing messages in a database table

The loaded send feature uses a property file (`loadedsend.props`) where the maximum and minimum number of messages that can be maintained are specified. This file can be placed anywhere. The server should be started with the following system property specifying the path to the properties file:

```
-Dcom.pramati.loadedsendprops = <path>/<to>/<the>/<loadedsend.props file>
```

Here is how the `loadedsend.props` property file looks:

```
queues=JMSQueue,SellQueue
minruntimesize=100
maxruntimesize=2000
```

In the above:

- `queue`: Refers to the name of the queues using the loaded send feature, specified as comma separated values.
- `minruntimesize`: The minimum number of messages maintained in-memory. Messages are loaded from the temporary table when the number of messages go below the minimum size as specified in the `loadedsend.props` file.
- `maxruntimesize`: The maximum number of messages maintained in-memory.

Messages in the loaded send operations are stored in a temporary table. Below is the schema for the loaded send temporary table. A temporary table (`temp_messages`) should be created comprise of the following columns:

- `mid`: The message id for the message. A varchar datatype of size 254.
- `message`: The actual message. A long datatype.
- `destination`: The destination to which the message is to be sent. A varchar datatype of size 60.
- `expirationtime`: The time after which the message expires (specified in milliseconds). A long datatype.

Note that this table must be created in the database to which the JMS server uses to store messages. Here are a few examples:

- Cloudscape:

```
CREATE TABLE temp_messages (mid varchar(254) primary key,
                            message long bit varying,
                            destination varchar(60),
                            expirationtime numeric(32));
```

- Oracle:

```
CREATE TABLE temp_messages (mid varchar(255) primary key,
                            message long raw,
                            destination varchar(60),
                            expirationtime number);
CREATE INDEX temp_messages1 ON temp_messages(destination)
```

Starting Standalone Message Server using Command line

1. Open a command window at <install_dir>/server/bin
2. Execute: `runserver -node <nodename>`
3. Specify the Message Server node name in <nodename> that you have created.

Note: *For a Message Server node, ensure that JMSService is enabled in server-config.xml, located at [server_install]/server/nodes/<node_name>/config.*

The command to start Standalone J2EE Server, a J2EE Server Cluster or Message Server node is same.

The command `runserver -node <nodename>` also helps in creation of a new node.

Understanding Message Server Configuration

All the above operations can be performed by changing the default configurations in the JMS configuration file jms-config.xml. This section provides an overview of how the configuration file can be modified for configuring a standalone Message Server.

Configuring Large Message Handling

A JMS message is termed as 'large message' when the message exceeds the maximum size that the server can handle. This maximum size is configured on each destination separately. The maximum size of the message is specified through the following tag under each destination.

```
<max-message-size-kb>50</max-message-size-kb>
```

Also, you can specify the action for the Server, when the message exceeds the maximum size. by configuring the <message-size-overflow-policy> value as follows:

```
<message-size-overflow-policy>
     handle-with-persistence
</message-size-overflow-policy>
```

The value can be either be 'handle-with-persistence' or 'throw-exception'.

If the policy is 'handle-with-persistence', when the sender sends a large non-persistent message, then the server will override its delivery mode from persistent to non-persistent and continue sending the message to the receiver. If the policy is 'throw -exception', then when the sender sends large message,

The default max-message-size-kb is *50 kb*. The default value for the over flow policy is 'handle-with-persistence'. The maximum size of each message that will be kept in memory can be configured using the max-message-size-kb property:

```
<max-message-size-kb>50</max-message-size-kb>
```

All the above mentioned default values can be changed in <default-destination-properties> of <admin-params>.

If you don not specify the max-message-size-kb at destination, then the destination will take the value that is specified at default-destination-properties. For temporary destinations you cannot configure separately. It will take the value which is specified in the default-destination-properties.

Configuring JMS Persistence Framework

JMS Message server requires a local store for storing its persistent messages. The messages can be persisted in two ways:

1. Database.
2. File System.

The Server provides a way of configuring this persistence type by exposing certain properties in the jms-config.xml file. If your system does not involve handling any persistent message, you don't need to configure any value.

With the following tags, we can configure persistence in JMS Server. The tag is part of the <admin-params> of the jms-config.xml:

```
<persistent-store>
      <persistence-enabled>true</persistence-enabled>
      <!-- persistence-disabled-policy: throw-exception,
      override-message- delivery-mode(default) -->

      <persistence-disabled-policy>
            override-message-delivery- mode
      </persistence-disabled-policy>

      <!-- persistence-type: file-store(default), db-store -->

      <persistence-type>
            db-store
      </persistence-type>

      <file-store>
            <!-- messages-dir is the absolute path or
            relative with prefix $NODES_DIR/ or use prefix $INSTALL_DIR/
            to make the store relative to install dir
            -->
            <messages-dir>$NODES_DIR/jmsmessages</messages-dir>
            <max-file-size-mb>10</max-file-size-mb>
            <purge-interval-sec>600</purge-interval-sec>
      </file-store>
      <db-store>
            <data-source-name>jmsdb</data-source-name>
            <jms-table-name-prefix>pmt_</jms-table-name-prefix>
      </db-store>
</persistent-store>
```

If you do not want to persist messages, then you can explicitly disable the persistence by specifying the <*persistence-enabled*>*false*</*persistence-enabled*>. If this flag is set then the message server does not check for any other configuration for the persistence. Whenever you set this flag as false, you also specify the <*persistence-disabled-policy*>, which denotes what to do if the persistence is disabled and the user is sending persistent messages. The persistence disabled policy should be either 'override-message-delivery-mode' or 'throw exception'. If the persistence-disabled-policy is 'override-message-

delivery-mode', then if the server receives any persistent messages, then it overrides the delivery mode from persistent to non-persistent. If the policy is 'throw exception', then for each persistent message that it receives, it will throw the exception indicating that the persistent messages are not allowed when persistence is disabled.

You can specify the persistence store type by using the tag *<persistence-type>*, which accepts either file-store or db-store.

Configuration for file-store:

```
<file-store>
     <messages-dir>$NODES_DIR/jmsmessages</messages-dir>
     <max-file-size-mb>10</max-file-size-mb>
     <purge-interval-sec>600</purge-interval-sec>
</file-store>
```

You have to specify in which directory you want to store the messages. This can be specified in multiple ways. You can either specify the absolute path of the directory, or you can specify $NODES_DIR/, which stores the messages under server's nodes directory with the child directory called messages or you can also specify it as $INSTALL_DIR/ which stores the messages under installation directory with the child directory called messages.

The file, stores the actual message objects. You can configure the *max-data-files size* in kilo bytes. So, if the first data file reaches to the configured max size, then it creates a new file and stores the data in it. The messages in the file store are actually not removed from the file and should be deleted as when required. There is a background thread which removes the messages as marked for deletion. So, this thread will wake up at configured interval which you specified through *<purge-interval-sec>*.

Configuration for DB store:

```
<db-store>
     <data-source-name>jmsdb</data-source-name>
     <jms-table-name-prefix>pmt_</jms-table-name-prefix>
</db-store>
```

Database configuration requires a datasource name, which is bound as a resource in naming service. For further details about how to configure the data source in resource-config.xml, check out the *Installation Guide*.

You can specify the *<jms-table-name-prefix>* for the tables that the server is going to create in the specified database. When the server starts up it automatically creates the required tables with the given prefix. The default values are:

Table 18: Persistence Framework Default Values

Property	Default Value
Persistence	Enabled

Table 18: Persistence Framework Default Values

Property	Default Value
Persistence Type	File Store
Message Dir.	$NODES_DIR/messages
Maximum File Size	10KB
Purge Interval	600 sec

Starting a Server Node Using Command line

The command to start Standalone J2EE Server, a J2EE Server Cluster or Message Server node is same.

To start a Standalone Message Server using Command line:

1. Open a command window at <install_dir>/server/bin
2. Execute: `runserver -node <nodename>`
3. Specify the Server node name in <nodename> that you have created.

Note: *For a Message Server node, ensure that JMSService is enabled in server-config.xml, located at [server_install]/server/nodes/<node_name>/config.*

The command `runserver -node <nodename>` also helps in creation of a new node.

Configuring Message Server Cluster Using Server Admin Service

Pramati Server provides the following ways to configure Message Server Cluster:

- Using the Server Administration Service
- Modifying XMLs

Before configuring the cluster, the Message Persistence tables should be created. Run the scripts in <install-dir>/jms/templates. You can modify the table names in the scripts to match the names specified for the four message persistence tables. Table names should be unique to a Message Server cluster to avoid conflicts among various standalone Message Servers and clusters since the same tables will be used by all the nodes of the cluster.

The following tables store runtime information required to persist messages.

- **Message table** -This table stores all persistent messages.
- **Subscriber table** -This table stores information about all subscribers.
- **Subscriber Message table** -This table stores messages of all subscribers.
- **XA table** -This table stores information about messages produced/consumed as part of distributed transactions.

Configuring Message Server Cluster

To configure a Message Server cluster using Server Administration Service:

1. Start the Server Administration Service.
2. Click 'Message Server Clusters' in the Explore panel. This displays 'Step 1 of 2: Add Message Server Cluster' in the Display panel. Provide details for the following fields.

Table 19: Step 1 of adding a Message Server cluster

Fields	Options	Description
Name		This is the name of the Message Server cluster. Say, 'MyJMS'.
Database		This contains mandatory fields indicating the database table that will hold cluster configuration details.
	URL	This is the URL of the database. You can edit the default text (jdbc:oracle:thin:@<CLUSTER_DB_HOST>:1521:<CLUSTER_DB_SID >) to point to the database you want. For Oracle, it can be jdbc:oracle:thin:@db1.pramati.com:1521:db1.
	Driver	This is the driver needed to connect to the database. You can edit the default text, oracle.jdbc.driver.OracleDriver.

Table 19: Step 1 of adding a Message Server cluster

Fields	Options	Description
	User Name	This is the username used to access the selected database. For example, if the database selected is Oracle, the user name can be 'scott'.
	Password	This is the password required to access the selected database. For example, if the database selected is Oracle, the password can be 'tiger'.
	Table	This is the name of the table in the database which will hold cluster configuration information. Say, 'emp'.
Message Persistence		In the Message Persistence area, you need to supply the following mandatory message persistence information.
	URL	This is the URL of the database. You can edit the default text (jdbc:oracle:thin:@<CLUSTER_DB_HOST>:1521:<CLUSTER_DB_SID >) to point to the database you want. For oracle, it can be jdbc:oracle:thin:@db1.pramati.com:1521:db1.
	Driver	This is the driver needed to connect to the database. You can edit the default text, oracle.jdbc.driver.OracleDriver.
	Username	This is the username with which you can access the selected database.
	Password	This is the password required to access the selected database.
	Message Table	This is the table name where all the messages will be stored. By default, the name displayed is 'messages'.
	Subscriber Table	This is the table name where all the subscriber details will be stored. By default, the name displayed is 'subscribers'.
	Subscriber Message Table	This is the table name where all the pending messages for durable subscribers are stored. By default, the name displayed is 'pending_messages'.
	XA Table	This table stores information about messages produced/consumed as a part of distributed transactions. Say, 'xa_messages'.

Click 'Next >>'. This displays 'Step 2 of 2: Add Message Server Cluster' in the Display panel.

Adding node(s) to Message Server Cluster

The next page enables creation and addition of nodes to the configured cluster.

In the 'Step 2 of 2: Add Message Server Cluster' page, you can create and add upto ten nodes at a time. For each created node, you need to specify the following node information:

Table 20: details needed for Adding a Message Server cluster node

Fields	Description
Name	This name identifies the node in the configured Message Server Cluster. Say, 'MyJmsNode'.

Table 20: details needed for Adding a Message Server cluster node

Fields	Description
Host	This is the name of the machine where the node will be configured (say, 'localhost' or '192.168.1.12'). The node can be configured on any machine in the local LAN where Administration Service is running.
Lookup Port	The port on which the node's Naming Service will be started. Say, '9191'.

Click 'Next >>'. A successful addition of the cluster takes you to the Server Administration Service Home page, displaying the added Message Server Cluster.

Starting Message Server Cluster

On the Server Administration Service Home page, select the Message Server Cluster you want to start.

1. Click 'Start' against the required Message Server cluster node. This displays details of the nodes that are defined for the cluster along with an option to specify JVM options, and the Security details of the user.
2. Select all the nodes that you want to start.
   Note: *You can click on the name of the node to view the node details.*
3. The Default security realm is '*system*', and the default username is '*root*'. Enter the password for the relevant realm. The default value for the '*system*' realm is *pramati*.
4. Click 'Start'.

A successful start takes you to the Server Administration Service page.

If you select a single node, or did not select all the nodes, you can see an option to 'Start other nodes'.

Note*: To start a Standalone Message Server or Message Server Cluster with a Standalone J2EE Server or J2EE cluster, the J2EE nodes must be started with their <jms-service> option.*

Connecting to Cluster

To connect to the Cluster:

1. Click 'Connect' against the Message Server cluster. The login page appears.
2. Enter the user name and password for the realm. For the *system* realm, the default values are *root* and *pramati* respectively.
3. Click 'Login'.

You are connected to the selected node.

The name of the connected node is displayed as selected in the Explore panel, while the Display panel shows details regarding alerts and the connected node.

Viewing Message Server node details

To view details about a Message Server node, click 'Configure' against the Message Server Cluster name on the Server Administration Service Home page. This displays the 'Home > Message Server Cluster' page, displaying all the nodes of the selected Message Server Cluster.

In the Nodes section, click on the name of the selected node whose details are to be viewed.

The 'Home > Message Server Cluster >Node' page displays information for the node such as name, Host and Lookup port.

Modifying Message Server Cluster configuration

You can modify the Lookup port details for a node by clicking 'Configure' in the Nodes section of 'Home > Message Server Cluster' page.

To configure a node:

1. Click the name of the cluster in the Message Server Clusters area of Server Administration Service Home page.
2. This displays the cluster configuration details that were defined while creating the cluster along with the list of nodes configured for that cluster.
3. To modify the Lookup port details of a node, click on 'Configure' corresponding to the desired node.
4. In the 'Home > Message Server Cluster Node' page, edit the Lookup port field and click 'Save'. The Status panel displays a success message if the change in configuration is successful.

Stopping Message Server node

When a cluster is started and is running, the corresponding row is highlighted in green in the Server Administration Service Home page.

To stop a specific node, connect to URL:

http://<IP of the machine running adminservice>:4748/admin?jndiUrl=rmi://<IP of the node>:<naming port of the node>

This opens the Console screen displaying details for the nodes. Click 'Stop'. You are prompted if the node is to be stopped. Clicking 'OK' stops the same, and displays an appropriate message.

On the Server Administration Service Home page, click 'Refresh'. The Message Server cluster area now displays the updated list of nodes.

Stopping Cluster

To stop a Cluster, use the 'Stop Cluster' option provided on the Console. You are prompted if the node is to be stopped. Clicking 'OK' stops the same, and displays an appropriate message.

You need to explicitly click on the 'Refresh' button provided on the Server Administration Service page to display the current status of the cluster. Once the cluster is stopped, a checkbox is made available for deletion of the same.

To stop a connected node of a Cluster, use the 'Stop' option provided after selecting the node on the Console.

Deleting Cluster

You can delete a cluster only when it's status is stopped.

To delete a cluster, select it using the checkbox provided, and click 'Delete'.

Running client for Message Server Cluster

A Message Server client can use a Message Server cluster in the same way as a Standalone Message Server, except that the client is required to supply the URL of one of the running Message Server cluster nodes, as "`java.naming.provider.url`" property in the client's properties file/object, `jndi.props`. The `jndi.props` file is used to create an InitialContext while looking up administered objects.

If the client specifies the URL of only one node in the properties file, that node should be running when the client connects. There should be at least two nodes running for failover to occur. If the second node comes up after the client connects, the client will not have any information of this node and will not failover as the client gets informed of the participating nodes only while creating the IC.

The client can specify a composite URL - URLs of two or more Message Server cluster nodes separated by commas. This ensures fail-safe acquisition of ICs so that the client can connect to any running node.

Say, `rmi://192.168.1.29:2001,rmi://192.168.1.29:2002`

Other properties are similar to those specified for a Standalone Message Server:

- Value for `java.naming.factory.initial` is `com.pramati.naming.client.PramatiClientContextFactory`
- `java.naming.provider.url` is the URL
- `java.naming.security.principal` is <user-name>, and the default value is '*root*'
- `java.naming.security.credentials` is <password>, and the default value is '*pramati*'
- `com.pramati.naming.realm` is <realm-name>, and the default realm is '*system*'

Managing Cluster Nodes

A cluster is a combination of individual instances of server known as nodes. Each node has its own set of replicated and non-replicated services. Pramati Cluster supports these node types:

Table 21: Cluster types supported by Pramati Server

Node Type	Description
BOTH	If a cluster node is defined as BOTH, both EJB and Web containers are started. Select this option when the application has Enterprise beans and Web components.

Table 21: Cluster types supported by Pramati Server

Node Type	Description
Web	If a cluster node is defined as Web, only the Web container is started. EJB components are not available on this node. Select this option when the application has only JSPs and other Web components. A standalone Web node can act as a EJB Load Balancer.
EJB	If a cluster node is defined as EJB, only the EJB container is started. Web components are not available on this node. Selection this option when the application consists of only enterprise beans.
JMS	Selecting the node type as JMS leads to the creation of a Message Server cluster.

You can add a Message Server cluster node through the Web interface only if:

- a host already exists on the network
- the local or any other machine on the network is the host
- the host is a remote machine, the start-up service should be running on the remote machine
- Lookup and HTTP port combination should be unique in the cluster
- Name of the node is unique to the host

Creating and Configuring Pramati Server Nodes Using XML

Overview

Pramati Server instances can be configured manually by modifying the configuration XML files. Server instances of type J2EE, Message Server and Cluster can be manually created with simple configuration in the XML files. This section describes the structure of a Server instance and discusses the procedure to create different types of Server instances in Standalone and Cluster scenarios.

Understanding Pramati Server nodes

An application server node can serve either as a Standalone server or a cluster server node. Pramati Server nodes can either be a Standalone Server or Server Cluster Node.

When the server starts, the server node is spawned as a separate process with its own JVM. Attributes of the node—sizing, runtime behavior and performance—is defined by the configuration of the server and is specified in a set of server configuration files (XMLs).

Key terminology

The following table provides a brief description for the various terms used in this document.

Table 22: Terminology

Term	Description
Installation	Pramati Server product Installation
Install-Dir	The directory under which Pramati Server is installed
Nodes Directory	The directory under which all configured server instances exist
Server	The installed server with default settings
Server Instance	An instance of a Pramati Server- of any type (J2EE, Web cluster node, EJB cluster node, Message Server) will normally run in one VM.
Standalone Server	An instance that is a self contained J2EE Server running in a single VM
Instance Directory	The directory under the *Nodes Directory* that holds the configuration and other files/information for a Server Instance
Message Server	An instance which functions as a Standalone Message Server
Embedded Message Server	Full featured Message Server that runs *inside* a Standalone Server instance
JMS Adapter	JMS Adapter is setup in a J2EE Server instance determining the connectivity to a Message Server (Embedded or external Standalone Message Server or HA-Message Server)

Table 22: Terminology

Term	Description
Cluster	A logical server comprising a group of underlying nodes forming a single entity that internally works with a collection of nodes.
Cluster Node	Server instance that serves as a node of a cluster
J2EE Cluster	A cluster with both EJB and WEB Nodes
J2EE Node	A Server instance that serves as a node serving both Web and EJB requests. Can exist as a part of aJ2EE Cluster
BOTH Node	Same as J2EE Node
Web Cluster	A cluster of Web Nodes
Web Node	A Server instance that serves as a node serving Web requests alone. Can either exist as part of a J2EE Cluster or a Web cluster
EJB Cluster	A cluster of EJB Nodes
EJB node	Server instance that serves as a node serving EJB requests alone. Can either exist as part of a J2EE Cluster or a Web cluster
Message Server Cluster	A cluster of Message Server Nodes
HA Message Server	Message Server Cluster that only supports failover, and not Load Balancing (as of 4.1, Message Server Cluster is only HA Message Server)
Message Server Node	A Server instance that serves as a node serving Message Server requests alone
Load Balancer	A request dispatcher for managing requests
Configuration	Modifying the default settings for the server

Pramati Server configuration files

Configuration files required for setting up Pramati Server are located in the /templates directory under the installation root. These files, with their descriptions, are listed in the following table.

Table 23: Pramati Server configuration files

Configuration File	Description
server-config.xml	The main configuration file where all the details related to services bootstrapping is stored. When you create a node manually, make sure that you copy this file in the config directory from the template.
resource-config.xml	All the resource related information like Connection Factory, Data Source properties, Data Source Cluster, Mail Resource and Message Server Resource are stored here.
web-config.xml	Necessary configuration information for the Web Container Service is stored here.
deploy-config.xml	Contains configuration information for the deploy service.

Table 23: Pramati Server configuration files

Configuration File	Description
logging-config.xml	Logging parameters like rotation type, max file size and rotation time are configured here.
security-config.xml	Contains information about security providers, security policy, certification manager and realm.
web-cache.xml	Contains dynamic cache configuration. Dynamic Cache service boosts the network throughput and access speed in the web container of Pramati Server.
cluster-config.xml	Stores cluster related information including configuration details for individual nodes.
jms-config.xml	Information related to Message Server Queues and Topics are stored here.

Pramati Server instances that can be configured

There are three basic types of Pramati Server configurations:

- J2EE (serves EJB and web applications)
- Message Server (serves pure Message Server applications)
- Load Balancer (acts as the Pramati Request Dispatcher)

These are set up in the configuration XML, `server-config.xml`. The default instance is J2EE. In a J2EE configuration, the Server can be set up as a Standalone Server or a Server Cluster Node. In a Message Server configuration, the Server can be set up as a Standalone Server or a High Availability Node.

Note: *In the J2EE configuration, the Server can run Message Server in an embedded mode.*

From these three basic configuration types, the following Server instances can be set up:

Standalone Server configurations derived from an instance

- Standalone Server
- Standalone Server with embedded Message Server
- Standalone Message Server

The most basic type of J2EE Server instance is the Standalone Server. It exists on a single machine, is not part of any cluster, and acts as an independent server. The Standalone Server is the default server type and can be configured to run embedded Message Server.

Note: *By default embedded Message Server is checked as enabled in a J2EE server instance.*

Server Cluster Node configurations derived from an instance

- J2EE Node (termed BOTH Node)
- Web Node
- HA Message Server Node

- EJB Node
- Load Balancer Node

The Server Cluster J2EE Node has both Web and EJB services enabled in `server-config.xml` file. A node can be configured to as a Standalone Message Server by disabling other services like EJB and Web—as long as there are no inter-service dependencies. The type of server instance is determined based on the specific services that are enabled through the Pramati Services Framework. Details are discussed in the following sections.

Structure of Nodes Directory

Creating a Server instance (in the node directory) makes a directory each for Standalone Server and Server Cluster Node.

Each configured server instance, has an unique name by which it is identified in a cluster. Instances can be created and configured manually without the aid of the Console or Shell.

While creating server instances (in the node directory), there will be one DIR for each server instance (in case of Standalone servers) and one for each cluster-node in case of Cluster. The type of server instance is determined based on the specific services enabled. Details are discussed in the following sections.

The installation directory structure

/server

-bin

(Server startup scripts are stored here)

-lib

(External and internal libraries used by the server are stored here)

-templates

(This directory contains various configuration XMLs and their DTDs along with the cloudscape DB property file)

-nodes

(Server Instances Directory containing node specific configuration information)

Server Instance Directory Structure

-<node-name>

(A unique name which identifies the node in a cluster)

-config

(List of all configuration XML files which influences the runtime behavior of the node.)

-logs

(Stack traces and log messages for the node are stored here)

-archives

(This directory holds all extracted application archives

-temp

(A temporary directory)

Procedure for creating Pramati Server Instances

Server instances are created in the 'nodes' directory of a Pramati Server installation. There will be one DIR for each server instance (in case of Standalone servers) and one for each cluster-node in case of Cluster. The type of server instance is determined based on the specific services enabled. Details are discussed in the following sections.

The common steps involved in configuring a Server instance, of any type, are:

1. Under the nodes directory, create a sub-directory (henceforth referred to as *instance-directory*) for the required instance.
2. Name of the sub directory will be the name of the instance.
3. Create the following sub directory under this instance's directory

   - *config*

   - the other directories described in the 'Structure of Nodes Directory' above will be automatically created after the instance starts up.
4. Depending on the type of instance needed, copy the required config files from the <install_dir>/server/templates to the configuration directory created above. The specific files needed are listed in the Table 3 (*Configuration files needed for Server Instances*) provided in the following section.
5. Edit the required configuration XML to setup the information as described in the following sections, relevant to the type of INstance being setup.
6. The instance can be started from the command line using the *-node* option, after running setup.bat /sh:

```
com.pramati.Server -node <node name>
```

Configuration files:

The configuration files determine the type of Server instance and the behavior of the server instance. For each server instance type a specific set of configuration files are required. The table below lists the config files required for each Instance type. These will need to be copied from templates directory into the <instance>/config directory.

Configuration File Dependency

Table 24: Configuration files needed for Server Instances

Configuration File	Standalone Server	J2EE (BOTH) Cluster Node	Web Cluster Node	Standalone Message Server	HA Message Server Node	EJB Cluster Node	Load Balancer
server-config.xml	Required	Required	Required	Required	Required	Required	Required
resource-config.xml	Required	Required	*Can Have*	Required	Required	*Can Have*	-
web-config.xml	Required	Required	Required	-	-	-	Required
deploy-config.xml	Required	Required	Required	-	-	Required	-
logging-config.xml	*Can Have*	*Can Have*	*Can Have*	*Can Have*	*Can Have*	*Can Have*	*Can Have*
security-config.xml	Required	Required	Required	Required	Required	Required	Required
system-security.xml	Required	Required	Required	Required	Required	Required	Required
web-cache.xml	Required	Required	Required	-	-	-	Required
cluster-config.xml	-	Required	Required	-	Required	Required	Needed, if HA
jms-config.xml	*Needed for embedded Message Server*	-	-	Required	Required	-	-
web-lbconfig.xml	-	-	-	-	-	-	Required

Note: *In most cases, the defaults used in the configuration file templates will be sufficient to start an instance.*

Creating a Standalone Server

General Instructions:

- Under the nodes directory, create a sub-directory for the required instance.
- Name of the sub-directory will be the name of the instance.
- Create the following sub-directory under this instance's directory

- config

- the other directories described in the '*Structure of Nodes Directory*' above will be automatically created after the instance starts up.

- Depending on the type of instance needed, copy the required config files from the INSTALL-DIR/server/templates to the configuration directory created above. The specific files needed are listed in Table 2 provided in the following section.
- Edit the configuration XML to setup the information as described in the following sections.
- Make sure that the cluster service is disabled in `server-config.xml`

```
<service name="ClusterService" enabled="false"
       class-name="com.pramati.cluster.PramatiClusterService">
```

- The instance can be started using *-node* option.

A Instance can be created by just creating a directory with an unique name under *server/nodes* directory. Once the directory is created, create another directory called *config* and copy `server-config.xml` and other configuration files required for the Standalone Server (as described in Table2*)* from the templates directory to *config*. Once the node directory is created, you can activate the Instance using *-node* command.

The same process can be followed for all the instance types including Message Server instances.

Once the Standalone J2EE Server is setup, based on the type selected, setup the basic attributes of the Web and EJB Containers. In the `server-config.xml`, change the values as needed:

```
<service name="WebContainer" enabled="true"
       class-name="com.pramati.web.WebServer">
              <config-file>web-config.xml</config-file>
              <requires always="NamingService,DeployService" />
              <property name="http-port" value="8181" />
              <property name="https-port" value="443" />
              <property name="ssl-enabled" value="false" />
</service>
```

Default value of HTTP port, and HTTPS port can be set now.

For EJB, in `server-config.xml`, the default pool size and session time-out value can be modified.

```
<service name="EJBContainer" enabled="true"
       class-name="com.pramati.ejb.EJBContainer">
       <requires always="NamingService,TransactionService,
              DeployService" />
       <property name="default-entity-min-pool-size" value="40" />
       <property name="default-entity-max-pool-size" value="1000" />
       <property name="default-session-min-pool-size" value="40" />
       <property name="default-session-max-pool-size" value="1000" />
       <property name="default-mdb-min-pool-size" value="40" />
```

```
        <property name="default-mdb-max-pool-size" value="1000" />
        <property name="default-low-activity-interval" value="60" />
        <property name="default-session-timeout" value="1000" />
        <property name="smart-code-generation" value="true" />
</service>
```

Enabling embedded Message Server in a Standalone Server

For enabling embedded Message Server for a Standalone Server follow the general instructions as given above for creating a Standalone Server. Add the following:

```
<service name="JMSService" enabled="true"
        class-name="com.pramati.jms.server.PramatiMessageService">
        <config-file>jms-config.xml</config-file>
        <requires always="NamingService,ResourceService"
        if-enabled="TransactionService" />
</service>
```

If it is not already present in your `server-config.xml` file. By default the standalone server is configured to start embedded Message Server service on services framework startup. `enabled="true"` lets the server start the Message Server Service. For an embedded Message Server, both EJB and Web container can be enabled in `server-config.xml`.

```
<service name="WebContainer" enabled="true"
        class-name="com.pramati.web.WebServer">
service name="EJBContainer" enabled="true"
        class-name="com.pramati.ejb.EJBContainer">
```

Restricting services in a Standalone J2EE Server

Based on the Pramati services framework, the services can be disabled or enabled. The services include the core containers (Web and EJB) and other services such as Resource and Transactions. A Standalone J2EE Server by default has both Web and EJB Container services enabled:

```
<service name="WebContainer" enabled="true"
        class-name="com.pramati.web.WebServer">
<service name="EJBContainer" enabled="true"
        class-name="com.pramati.ejb.EJBContainer">
```

For a Web only J2EE Server, enable the Web Container Service and disable the EJB Service.

```
<service name="WebContainer" enabled="true"
        class-name="com.pramati.web.WebServer">
<service name="EJBContainer" enabled="false"
        class-name="com.pramati.ejb.EJBContainer">
```

For an EJB only J2EE Server, enable the Web Container Service and disable the EJB and Message Server Services.

```
<service name="WebContainer" enabled="false"
```

```
        class-name="com.pramati.web.WebServer">
<service name="EJBContainer" enabled="true"
        class-name="com.pramati.ejb.EJBContainer">
```

Creating Standalone Message Server

For a Standalone Message Server, enable the JMS Service in the `server-config.xml`:

```
<service name="JMSService" enabled="true"
        class-name="com.pramati.jms.server.PramatiMessageService">
        <config-file>jms-config.xml</config-file>
        <requires always="NamingService,ResourceService"
        if-enabled="TransactionService" />
</service>
```

Disable the Web Container and EJB Container services from `server-config.xml`:

```
<service name="WebContainer" enabled="false"
        class-name="com.pramati.web.WebServer">
<service name="EJBContainer" enabled="false"
        class-name="com.pramati.ejb.EJBContainer">
```

Customizing Server Footprint

The Server framework allows customizing a server instance to the extent of choosing the specific set of services required by the application. This will result in just those services being started. The binary files (Jar) for the remaining services can even be removed from the Server install directory. Services can be enabled and disabled by changing configuration in `server-config.xml`. More details are dealt in the following section.

Using Pramati Server NodeCreator

The NodeCreator is a utility for creating or removing nodes and clusters. You can configure and use the NodeCreator with both the command line and XMLs.

Using NodeCreator with Command line

To run the NodeCreator:

Run NodeCreator.bat (for Windows) or NodeCreator.sh (for Unix) located at <install_dir>/server/bin/.

Execute java `com.pramati.NodeCreator` class by passing the following values:

Table 25: Operations that can be performed using the Node Creator class

Operation	Description
-createj2ee	To create a Standalone J2EE node
-createj2ecluster	To create multiple J2EE nodes in a cluster
-createjms	To create a Message Server node
-createjmscluster	To create multiple Message Server nodes in a cluster

The various values (given as {parametername value} format) required to perform the above operations are:

Table 26: Values for the Node Creator class

Value	Description
-name	Name(s) of the nodes. In case of a cluster, the names are separated by a comma ','
-ip	The IP of the machine on which this node is being created. The default value is local host. In case of a cluster, the IPs are separated by a comma ','
-namingport	Naming port(s) of the nodes. In case of a cluster, the ports are separated by a comma ','
-httpport	HTTP port(s) of the nodes. In case of a cluster, the ports are separated by a comma ','
-clustername	In case of a cluster, the name of the cluster.

You can also specify the System Property '`install.root`', if the program is not being run from the Server root.

Example for creating a J2EE Standalone node

```
java -Dinstall.root=. com.pramati.NodeCreator -createj2ee -name my
node1 -namingport 9191 -httpport 8181
```

Example for creating a J2EE cluster

```
java -Dinstall.root=. com.pramati.NodeCreator -createj2eecluster -name my
node1,mynode2 -namingport 9191,9292 -httpport 8181,8282 -clustername myclus-
ter
```

The help snippet that shows up on the command shell is as follows:

```
Usage: java com.pramati.NodeCreator [operation] [option value] [flag]
```

Using NodeCreator with XMLs

You can create a node or cluster using the `config-topology-standalone-template.xml and config-topology-cluster-template.xml` located at <install_dir>/server/templates/.

Using these XML templates, you can convert existing node types, create nodes for a cluster, and configure persistence properties for the database, EJB or Web.

Converting a node type is useful if you want to convert an existing, say Standalone J2EE Server to an EJB cluster type. To do this, open the `config-topology-cluster-template.xml`:

```
<configuration-topology>
    <!-- Valid Types - a J2EE(EJB/BOTH/WEB) Cluster or a JMS Cluster-->
    <!-- SAMPLE CONFIGURATION, change node definitions as required -->
    <cluster name="MyCluster" type="J2EE">
        <convert-nodes>
<!-- These are nodes for conversion, the server "J2EESAS1" of the given ip in
the <source-node> tag would be converted to the cluster node type given in the
node definition. The target nodenames can be different in which case the
directories will be copied.-->
            <node name="EJB1" type="EJB1-cluster" naming-port="9000" http-
            port="8000">
                <source-node ip="localhost" name="J2EESAS1" />
            </node>
        </convert-nodes>
```

Table 27: Description of attributes for <convert-nodes> tag

Attributes	Description
node name	Name(s) of the target node. That is the node that is to be created.
type	The type of the target node that is to be created.
naming-port	Naming port(s) of the target node.
http-port	HTTP port(s) of the target node.
source-node ip	The IP of the source machine on which this node is started. The default value is local host.
name	Name of the source node.

If you wish to create a J2EE Cluster, locate the `<create-nodes>` tag in the `config-topology-cluster-template.xml`:

```
<create-nodes>
     <node name="nodename1" type="both">
        <ip>localhost</ip>
        <naming-port>9292</naming-port>
        <http-port> 8282</http-port>
     </node>
     <node name="nodename2" type="both">
        <ip>localhost</ip>
        <naming-port>9293</naming-port>
        <http-port> 8283</http-port>
     </node>
</create-nodes>
```

Table 28: Description of attributes for <create-nodes> tag to create a cluster

Attributes	Description
node name	Name of the cluster node.
type	The type of the cluster - EJB, Web, or BOTH.
ip	The IP of the machine on which this node is being created. The default value is local host.
naming-port	Naming port of the node.
http-port	HTTP port of the node.

To define the properties for persistence, modify the `<configuration-persistence>` tag in the `config-topology-cluster-template.xml`:

```
<!-- use "db" for DB persistence  -->
<configuration-persistence type="file">
    <property name="driver" value=""/>
    <property name="url" value=""/>
    <property name="user" value=""/>
    <property name="password" value=""/>
    <property name="tablename" value=""/>
</configuration-persistence>
<!--the default type here is "in-memory", use "db" for DB persistence -->
<ejb-cluster-persistence >
    <property name="driver" value=""/>
    <property name="url" value=""/>
    <property name="user" value=""/>
    <property name="password" value=""/>
```

```
        <property name="table-name" value=""/>
    </ejb-cluster-persistence>
    <!--the default type here is "in-memory", use "db" for DB persistence -->
    <web-cluster-persistence >
        <property name="url" value=""/>
        <property name="url" value=""/>
        <property name="user" value=""/>
        <property name="password" value=""/>
        <property name="table-name" value=""/>
    </web-cluster-persistence>
   <!-- the upload-config allows pre-configuring the created cluster.e.g.The
configuration for nodename1 would be uploaded
     to the DB and this configuration would be synced up to the other nodes.-
->
    <upload-config-source-node name="nodename1" />
</cluster>
</configuration-topology>
```

Using - configuration flag for custom node configuration

You can use the `-configuration` flag with the NodeCreator utility and pass the name of a configuration file as a parameter. For example: `com.pramati.NodeCreator -configuration config.xml`

The `config.xml` file can have the following structure:

```
<server-configuration>
  <configuration service-name="ResourceService">
    <datasource name="myDS" connection-factory="myCF" transaction-participa-
    tion="true" description="No Description">
      <pool-properties min-pool-size="10" max-pool-size="20" initial-pool-
      size="2" refresh-interval-seconds="" connection-request-timeout-sec-
      onds="" idle-timeout-seconds="" cache-size="" />
      <connection-validation sql="" class="" />
    </datasource>

  <connection-factory name="myCF" description="" classname="COM.cloud-
  scape.core.RmiJdbcDriver" url="jdbc:cloud-
  scape:rmi:D:\CTS\j2eects\db_cs40\DB1" authorized-by="Container">
    <login-parameters user="" password="" mask-password="false"/>
  </connection-factory>

  <-jms-adapter name="myAdapter" description="No Description" interface-
  class="com.pramati.jms.client.JMSProviderImpl">
```

```
      <properties>
       <property name="com.pramati.naming.cacheLookups" value="false"/>
       <property name="java.naming.factory.initial" value="com.pramati.nam-
       ing.PramatiLocalContextFactory"/>
       <property name="java.naming.provider.url" value="rmi://localhost:9090"/>
       <property name="com.pramati.force.standalone.ctx" value="false"/>
      </properties>
    </jms-adapter-->
  </configuration>
</server-configuration>
```

As an example, the above will add the mentioned resources in the newly created nodes - the relevant entries will be added to the `resource-config.xml` files.

The supported resources for this tag are:

1. JDBC Connection Factories
2. Datasources
3. JMS Adapters

Cluster Configuration using NodeCreator

Cluster configuration can be performed using the NodeCreator utility as shown in the following example:

Usage Objective

To have a cluster with four nodes (a1, a2, a3 and a4) on four different machines (n1, n2, n3 and n4) respectively.

Inputs: A sample standalone node containing all the resources and configurations. An XML file giving details of the cluster conversion.The template of the XML file can be found in $INSTALL_DIR/templates/config-topology-cluster-template.xml.

Requirements: To restore all the configuration of the sample node and modify some of the configuration as per requirement.

Steps Involved:

1. Start the Admin Service on each of the node where cluster nodes is required to be made. In this case (n1,n2,n3,n4). Admin Service can be started by using *runstartupsvc.bat* in $INSTALL_DIR/server/bin directory.
2. Use the NodeCreator utility to configure a cluster configuration. NodeCreator utility could be found in $INSTALL_DIR/server/bin directory. The command to use the nodecreator utility is

```
nodecreator -config_topology {$path_of_config_xml} -configuration
{$path_of_config_xml} -writetopology
```

The writetopology option creates the topology XML, which can be used to launch the configured cluster nodes. The topology XML is generated in the same directory as the config-topology

file. Use the nodecreator command without any parameters to see the help.

3 Use the node-creator utility to remove the configured cluster nodes. The command to use is `nodecreator -remove -topology {$path_of_topology_xml}.`

The topology XML is created while configuring the cluster if *-writetopology* option is used.

Configuring J2EE Server Cluster using NodeCreator

You can configure a J2EE Cluster using the NodeCreator class. To run the NodeCreator Utility:

Run `NodeCreator.bat` (for Windows) or `NodeCreator.sh` (for Unix) located at <install_dir>/server/bin/.

Execute java `com.pramati.NodeCreator` class by passing the following values:

Table 29: Operations that can be performed using the Node Creator class

Operation	Description
-createj2ee	To create a Standalone J2EE node
-createj2eecluster	To create multiple J2EE nodes in a cluster
-createjms	To create a Message Server node
-createjmscluster	To create multiple Message Server nodes in a cluster

The various values (given as {parametername value} format) required to perform the above operations are:

Table 30: Values for the Node Creator class

Value	Description
-name	Name(s) of the nodes. In case of a cluster, the names are separated by a comma ','.
-ip	The IP of the machine on which this node is being created. The default value is local host. In case of a cluster, the IPs are separated by a comma ','.
-namingport	Naming port(s) of the nodes. In case of a cluster, the ports are separated by a comma ','.
-httpport	HTTP port(s) of the nodes. In case of a cluster, the ports are separated by a comma ','.
-clustername	In case of a cluster, the name of the cluster.

You can also specify the System Property '`install.root`', if the program is not being run from the Server root.

Example for creating a J2EE cluster

```
java -Dinstall.root=. com.pramati.NodeCreator -createj2eecluster -name my
```

```
node1,mynode2 -namingport 9191,9292 -httpport 8181,8282 -clustername myclus-
ter
```

The help snippet that shows up on the command shell is as follows:

```
Usage: java com.pramati.NodeCreator [operation] [option value] [flag]
```

Creating LB Node using NodeCreator

Load Balancer node can be created using the NodeCreator utility as shown below:

```
java -Dinstall.root=. com.pramati.NodeCreator -createlb -name nodeA -naming-
port 9191 -httpport 8080 -topology <topology.xml>
```

In the above example a node named nodeA will be created and the node configuration is picked up from the topology.xml file created in the same directory as the config-topology-cluster-template.xml The attributes are explained in the following table:

Table 31: Attributes for creating LB Nodes

Attribute	Description
-name	Name of the cluster node.
-namingport	Naming port(s) of the nodes. In case of a cluster, the ports are separated by a comma ‘,’.
-httpport	HTTP port for the node.
-topology	Location to pickup the cluster topology and configuration.

Creating Pramati Server Clusters Using XMLs

Pramati Server features a powerful clustering solution. A cluster is defined as a group of nodes. Pramati Clustering solution offers transparent, self-organizing, homogenous web based configuration and management allowing easy EJB and Web clustering. When an application is deployed on any one of the cluster nodes, it is automatically replicated across all running cluster nodes. For a client, this translates to a single server view of the cluster.

Server clustering is self-organizing. Adding or removing nodes does not require restarting of the cluster. When a node is added, the load is automatically distributed across all the nodes based on their weight.

All the J2EE Services like Naming, Resource, Security and Transaction are automatically replicated across all the nodes across the cluster. Non-J2EE services like locking, application synchronization, load balancing, and failover are designed such that all complex cluster mechanics are hidden ensuring a single server view of the client.

Server clustering provides both EJB and Web clustering. Each Server node has its own set of replicated and non-replicated services. Pramati Cluster supports Web, EJB and BOTH Cluster-node types. Types of Clustering supported are:

- J2EE clustering
- Message Server clustering

Cluster Definition

Cluster definition involves:

- Defining a logical cluster: Comprises of defining the common configuration and state persistence information
- Defining the nodes of the cluster

The initial definition of the cluster is made in the XML of the first node of the cluster. Once the basic cluster attributes are defined in the `cluster-config.xml`. The cluster type is defined by the Cluster-node definition in the `cluster-config.xml` as shown in the following code snippet:

```
<node>
      <name>node</name>
      <computer-name>127.0.0.1</computer-name>
      <naming-port>9191</naming-port>
      <http-port>8181</http-port>
      <node-type>BOTH</node-type>
```

```
        <node-id>001</node-id>
    </node>
```

Here, when node-type is J2EE, enabling the cluster service would make it a J2EE Cluster. A J2EE Cluster can be of type BOTH (both ejb and web) or just EJB/Web.

J2EE Cluster cannot start with Message Server in an embedded mode and a Message Server cluster cannot have EJB or Web Containers enabled. If node-type is "JMS", then it becomes a HA Message Server cluster.

Once cluster is enabled in `server-config.xml`, define the cluster topology and properties in the `cluster-config.xml`. Specify cluster nodes, topology, persistence (EJB/Web) and Load Balancer here.

J2EE clustering can involve both EJB and Web clustering. However, Web and Message Server instances cannot exist in a single cluster. The cluster solution is based on a common cluster backbone called Cluster service, which also runs on the Pramati Services framework.

Pramati Cluster being a self-organizing, non master-slave homogeneous cluster, does not have a separate entity for cluster housekeeping or processing. Based on the shared persistence store, the cluster runtime in each cluster node jointly perform all the housekeeping and operational tasks.

Uploading Configuration to DB when DB is used for Configuration Persistence

To upload the configuration into the database in case of DB persistence for cluster configuration, start the node for the first time using the following command:

```
com.pramati.Server -node <NODENAME> -uploadconfig
```

The rest of the nodes would automatically pick up the configuration and do not need to be started with this option. To update the database after making changes in the configuration, start the cluster node with the following startup option:

```
com.pramati.Server -node <NODENAME> -updatelocalconfig
```

This option would update all of the configuration files. Only this node needs this option and the rest would automatically use updated configuration. To update specific configuration changed manually, start cluster node with the following startup option:

```
com.pramati.Server -node <NODENAME> -updatelocalconfigfor
<SERVICENAME>[,<SERVICENAME>]
```

Only this node needs this option and the rest would automatically use updated configuration.

Creating a Cluster Instance

The cluster definition is essentially defined along with the first cluster node, and the same common persistence information and cluster details should be setup in all the other nodes that form a part of this cluster. The steps involved in creating a cluster are:

1 The DIR and the config files should be setup just as one would setup a Standalone instance.

2 Define the name of the cluster. Open the `server-config.xml` file and add a declaration as follows:

```
<belongs-to-cluster> MyCluster </belongs-to-cluster>
```

3 Enable the cluster service. This cluster name is a logical representation for a group of nodes enabling them to communicate with each other. Once the node is identified as part of a cluster, the service definition for the cluster can be added in the `server-config.xml` file as follows:

```
<service name="ClusterService" enabled="true"
  class-name="com.pramati.cluster.PramatiClusterService">
  <config-file>cluster-config.xml</config-file>
  <requires always="NamingService"  />
</service>
```

Set the 'enabled' attribute to true, so the cluster service is started when the services framework is started. As you can infer from the above code snippet, the actual configuration file for the cluster is stored by every node in cluster-config.xml.

Note: *Though at present the nodes interact with each other through a repository of IP and port information, it is important that the nodes identify themselves with a common name.*

4 Setup persistence. The services configuration in a node can be stored either in a file or a database. The information can be specified in `server-config.xml` under `<configuration-persistence>` tag as shown in the code snippet below. By default all the configuration information are persisted to the file.

```
<configuration-persistence use="file">
  <persistence type="file" />
  <persistence type="db">
        <property name="driver" value="" />
        <property name="url" value="" />
        <property name="user" value="" />
        <property name="password" value="" />
        <property name="tablename" value="" />
  </persistence>
</configuration-persistence>
```

Persistence can be configured either at the file level or DB level. For DB level persistence, connection parameters like driver, connection URL, username, password and table name can be specified in the configuration file, which are persisted. A driver can be any valid JDBC driver and varies for different databases. The 'tablename' attribute is used to create a specific table where the configuration details for all the services are stored.

If you are setting up a new cluster, you can opt for a file level persistence, where the DB is not used for aiding a synch-up. The new version of Server supports an 'On Demand sync up', wherein the configuration information for each node are synched up with every other node and persisted in the DB.

Setup the cluster configuration file that would define all nodes that form part of this cluster. The file `cluster-config.xml` captures the information on the nodes that forms a cluster. Initially, when creating the cluster this information is to be provided in this file. This file will be identical in ALL the nodes that will form the cluster- with information on each one of them.

After the first startup of the server, the information gets persisted in the DB/File persistence store of the cluster, and a read-only version of the file is always available synched up from the DB on every 'start' of the cluster node.

1. Copy the `cluster-config.xml` file from the templates directory to the nodes/<node-name>/ config directory.
2. Add the node related information or topology for each node.

```
<node>
        <name>N1</name>
        <computer-name>128.128.77.77</computer-name>
        <naming-port>9191</naming-port>
        <http-port>8181</http-port>
        <node-type>BOTH</node-type>
        <node-id>001</node-id>
</node>
<node>
        <name> N2 </name>
        <computer-name> 128.128.77.78 </computer-name>
        <naming-port> 9190 </naming-port>
        <http-port> 8180 </http-port>
        <node-type> BOTH </node-type>
        <node-id> 002 </node-id>
</node>
```

Setting up a J2EE Cluster

Follow the general instruction for setting up a Cluster node as summarized here:

- Create the Node Directory and copy the configuration files required for the J2EE Node as mentioned in Table 2.
- Add cluster declaration in `server-config.xml`. Make sure that EJB and Web or EJB/Web are enabled and Message Server is disabled.

```
<service name="WebContainer" enabled="true"
    class-name="com.pramati.web.WebServer">
<service name="EJBContainer" enabled="true"
    class-name="com.pramati.ejb.EJBContainer">
<service name="JMSService" enabled="false"
    class-name="com.pramati.jms.server.PramatiMessageService">
```

- Enable Cluster Service.

```
<service name="ClusterService" enabled="true"
     class-name="com.pramati.cluster.PramatiClusterService">
```

- Setup Persistence.
- Setup Cluster topology in `cluster-config.xml`.

Note: *Make sure the node types mentioned in the node definitions in cluster-config.xml are of the type BOTH, Web or EJB. For more information, refer to 'Creating a Cluster Instance'.*

J2EE Cluster Node Types

The Cluster node types for both the nodes are indicated as BOTH, which means both the Web and EJB Services are running on that node. The node types allowed are:

- *BOTH* – If a cluster node is defined as BOTH, both EJB and Web containers are started. Select this option when the application has both enterprise and web components. If you specify BOTH, it is required that you enable both EJB and Web Services from the `server_config.xml` file as shown in the snippet:

```
<service name="WebContainer" enabled="true"
     class-name="com.pramati.web.WebServer">
<service name="EJBContainer" enabled="true"
     class-name="com.pramati.ejb.EJBContainer">
```

- *Web* – If a cluster node is defined as Web, only the Web container is started. EJB components are not available on this node. Select this option when the application has only JSPs and other web components. A Standalone Web node can act as an EJB Load Balancer. A snippet from `server_config.xml`:

```
<service name="WebContainer" enabled="true"
     class-name="com.pramati.web.WebServer">
<service name="EJBContainer" enabled="false"
     class-name="com.pramati.ejb.EJBContainer">
```

- *EJB* – If a cluster node is defined as EJB, only the EJB container is started. Web components are not available on this node. Select this option when the application consists of only enterprise beans. A snippet from `server_config.xml`:

```
<service name="WebContainer" enabled="false"
     class-name="com.pramati.web.WebServer">
<service name="EJBContainer" enabled="true"
     class-name="com.pramati.ejb.EJBContainer">
```

To summarize:

Table 32: J2EE Cluster Node Type and required services

J2EE Node Type	Web Container	EJB Container
BOTH	Required	Required

Table 32: J2EE Cluster Node Type and required services

J2EE Node Type	Web Container	EJB Container
EJB	-	Required
Web	Required	-

Advanced Configuration

Typically a J2EE Cluster node means a 'BOTH' node. But a J2EE Cluster can also be configured as a Web only cluster, EJB only cluster or a combination of Web and EJB nodes.

Setting up Web Replication and Persistence

You can set up a Web Cluster following the same procedure as shown in '*Creating a Cluster Instance*'. Make sure that you enable the Web Container service for all the nodes. This can be done from the individual `server_config.xml` files.

Additionally, you need to add the following information in your `cluster_config.xml` file:

```
<web-cluster>
      <enable>true</enable>
      <web-session-persistence>
            in-memory
      </web-session-persistence>
      <replication-on-all-nodes>
            true
      </replication-on-all-nodes>
       <web-cluster-persist-info>
            <url />
            <driver />
            <user />
            <password />
            <table-name />
      </web-cluster-persist-info>
</web-cluster>
```

Backup nodes can be setup for a Web Cluster only when the web session persistence is set to In-Memory and replication on all nodes is set to false. Enable the Web cluster by setting '*enabled*' as '*true*'. There can be two kinds of Web session persistence:

- In-memory Persistence
- Database Persistence

When you select your Web session persistence to be in-memory, the HTTP session objects are persisted in memory locally for nodes in the same VM and persisted through RMI calls for nodes in different VM.

Whenever there is any state change for Web sessions, the information is replicated across all other nodes. This may prove expensive on memory and hence it is better to set '*replication-on-all-nodes*' to '*false*'. When the '*replication-on-all-nodes*' are set to '*false*', only the backup nodes are notified for replication. The Web cluster persistence information can be stored in a DB.

Setting up EJB Persistence

EJB session persistence for J2EE Cluster could be in-memory or database based. In case of database-based persistence, provide all the database information like URL, table name, etc. The following snippet is taken from `cluster_config.xml`:

```
<ejb-session-persistence>in-memory</ejb-session-persistence>
    <persist-info>
       <url />
       <driver />
       <user />
       <password />
       <table-name />
     </persist-info>
```

The above example uses In-memory EJB session persistence.

Updating cluster configuration using XML (when DB is used for configuration)

Once cluster is enabled in `server_config.xml`, the cluster topology and properties are defined in the `cluster_config.xml`. Cluster nodes, topology, persistence (EJB/Web) and Load Balancer will be specified here. After the initial configuration of the cluster, the cluster configuration may be modified using the Console or the XMLs.

In a file based persistence, once the config XMLs are modified in any of the nodes, and the cluster is restarted, the new configuration takes effect and is synched-up with all other nodes.

When DB is used for config persistence, the following steps need to be followed to update the configuration in the DB, using local XMLs to provide the information:

- The cluster must be stopped
- In any of the nodes, update the required config XML
- Start the node, where the XMLs were modified:
  - After any change in the configuration, to update the database, start cluster node with the following option: `com.pramati.Server -node <NODENAME> -updatelocalconfig` This updates all of the configuration files.
  - To update specific configuration changed manually, start cluster node with the following startup option: `com.pramati.Server -node <NODENAME> -updatelocalconfigfor <SERVICENAME>[<SERVICENAME>]`

Only this node needs this option and the rest would automatically use updated configuration.

Configuring J2EE Server Cluster using XMLs

To configure a J2EE Cluster using XMLs use the following steps:

1. Create a new folder, say 'j2ee1' under the <install_dir>/server/nodes.
2. Copy the 'config' directory from <install_dir>/server/nodes/<server_name> and place it under <install_dir>/server/nodes/j2ee1. This copies the `server-config.xml`.
3. Copy the cluster-config.xml from <install_dir>/server/templates and place it under <install_dir>/server/nodes/j2ee1.
4. Open the `server-config.xml`, and make the following changes:

Table 33: Modifying the server-config.xml for configuring a J2EE Server Cluster

Tag with original value	Change value to
<server-node name="default" type="J2EE">	`<server-node name="j2ee1" type="j2ee">`
<!-- <belongs-to-cluster>MyCluster</belongs-to-cluster>-->	<belongs-to-cluster>MyJ2EECluster</belongs-to-cluster>. Note: The comment has to be removed.
<property name="namingport" value="9191" />	<property name="namingport" value="9191"/> Note: You can change the port value to the port being used.
<service name="DeployService" enabled="true"	Set enabled="false" if the Service is not needed
<service name="WebContainer" enabled="true"	Set enabled="false" if the Service is not needed
<service name="EJBContainer" enabled="true"	Set enabled="false" if the Service is not needed
<service name="JMSService" enabled="true"	Set enabled="false" if the Service is not needed
<service name="ClusterService" enabled="false"	Set enabled="true"
<configuration-persistence use="file">	<configuration-persistence use="db">
<property name="driver" value="" />	<property name="driver" value="oracle.jdbc.driver.OracleDriver"/>
<property name="url" value="" />	<property name="url" value="jdbc:oracle:thin:@qadb:1521:qadb"/>
<property name="user" value="" />	<property name="user" value="qa1"/>
<property name="password" value="" />	<property name="password" value="qa1"/>
<property name="tablename" value="" />	<property name="tablename" value= "mytable"/>

5. Open the `cluster-config.xml`, and make the following changes:

Table 34: Modifying the cluster-config.xml for configuring a J2EE Server Cluster

Tag with original value	Change value to
<name>default</name>	<name>j2ee1</name>

Table 34: Modifying the cluster-config.xml for configuring a J2EE Server Cluster

Tag with original value	Change value to
<computer-name>127.0.0.1</computer-name>	<computer-name>127.0.0.1</computer-name> Note: You can change the ID of your system here.
<naming-port>9191</naming-port>	<naming-port>9191</naming-port> Note: You can change the port value to the port being used.
<http-port>8181</http-port>	<http-port>8181</http-port>
<node-type>BOTH</node-type>	<node-type>BOTH</node-type> Note: The type can also be set to EJB or Web
<backup-node>node2</backup-node>	<backup-node>j2ee2</backup-node> Note: This is the name of the node that will be used in case of a failover. This node must also be configured in the same manner as 'j2ee1'.

Note: *The number of nodes created in the XML must be equal to the number of <new folders> created under <install_dir>/server/nodes.*

Deploying an application on Cluster node

Enter the following at the command prompt to deploy an application on J2EE Cluster:

```
j2eeadmin@clusternode1 deploy[path_to_app]
```

Configuring J2EE Server Cluster using NodeCreator

You can configure a J2EE Cluster using the NodeCreator class. To run the NodeCreator Utility:

Run `NodeCreator.bat` (for Windows) or `NodeCreator.sh` (for Unix) located at <install_dir>/server/bin/.

Execute java `com.pramati.NodeCreator` class by passing the following values:

Table 35: Operations that can be performed using the Node Creator class

Operation	Description
-createj2ee	To create a Standalone J2EE node
-createj2eecluster	To create multiple J2EE nodes in a cluster
-createjms	To create a Message Server node
-createjmscluster	To create multiple Message Server nodes in a cluster

The various values (given as {parametername value} format) required to perform the above operations are:

Table 36: Values for the Node Creator class

Value	Description
-name	Name(s) of the nodes. In case of a cluster, the names are separated by a comma ‘,’.
-ip	The IP of the machine on which this node is being created. The default value is local host. In case of a cluster, the IPs are separated by a comma ‘,’.
-namingport	Naming port(s) of the nodes. In case of a cluster, the ports are separated by a comma ‘,’.
-httpport	HTTP port(s) of the nodes. In case of a cluster, the ports are separated by a comma ‘,’.
-clustername	In case of a cluster, the name of the cluster.

You can also specify the System Property ‘`install.root`’, if the program is not being run from the Server root.

Example for creating a J2EE cluster

```
java -Dinstall.root=. com.pramati.NodeCreator -createj2eecluster -name my
node1,mynode2 -namingport 9191,9292 -httpport 8181,8282 -clustername myclus-
ter
```

The help snippet that shows up on the command shell is as follows:

```
Usage: java com.pramati.NodeCreator [operation] [option value] [flag]
```

Configuring Message Server Cluster Using XML

You can configure Message Server Cluster in Pramati Server in the following ways:

- Using the Server Administration Service. For details, read '*Configuring Message Server Cluster Using Server Administration Service*'
- Modifying XMLs as explained in this chapter.

Before configuring the cluster, run the scripts in <install-dir>/jms/templates to create the Message Persistence tables. You can modify the table names in the scripts to match the names specified for the four message persistence tables. Table names should be unique to a Message Server cluster to avoid conflicts among various standalone Message Servers and clusters since the same tables will be used by all the nodes of the cluster.

The following tables store runtime information required to persist messages:

- **Message table** -This table stores all persistent messages.
- **Subscriber table** -This table stores information about all subscribers.
- **Subscriber Message table** -This table stores messages of all subscribers.
- **XA table** -This table stores information about messages produced/consumed as part of distributed transactions.

Configuring Message Server Cluster using XML

To configure a Message Server cluster using XMLs use the following steps:

1. Create a new folder, say 'jmsc1' under the <install_dir>/server/nodes.
2. Copy the 'config' directory from <install_dir>/server/nodes/<server_name> and place it under <install_dir>/server/nodes/jmsc1. This copies the `server-config.xml`.
3. Copy the `cluster-config.xml` from <install_dir>/server/templates and place it under <install_dir>/server/nodes/jmsc1.
4. Open the `server-config.xml`, and make the following changes:

Table 37: Modifying the server-config.xml for configuring a Message Server Cluster

Tag with original value	Change value to
<server-node name="default" type="J2EE">	<server-node name="jmsc1" type="JMS">
<!-- <belongs-to-cluster>MyCluster</belongs-to-cluster>-->	<belongs-to-cluster>MyJMSCluster</belongs-to-cluster> <!--Note: The comment has to be removed.-->

Table 37: Modifying the server-config.xml for configuring a Message Server Cluster

Tag with original value	Change value to
<property name="namingport" value="9191"/>	<property name="namingport" value="9191"/> <!--Note: You can change the port value to the port being used.-->
<service name="DeployService" enabled="true"	Set enabled="false"
<service name="WebContainer" enabled="true"	Set enabled="false"
<service name="EJBContainer" enabled="true"	Set enabled="false"
<service name="JMSService"	Set enabled="true"
<service name="ClusterService" enabled="false"	Set enabled="true"
<configuration-persistence use="file">	<configuration-persistence use="db">
<property name="driver" value="" />	<property name="driver" value="oracle.jdbc.driver.OracleDriver"/>
<property name="url" value="" />	<property name="url" value="jdbc:oracle:thin:@qadb:1521:qadb"/>
<property name="user" value="" />	<property name="user" value="qa1"/>
<property name="password" value="" />	<property name="password" value="qa1"/>
<property name="tablename" value="" />	<property name="tablename" value= "mytable"/>

5 Open the cluster-config.xml, and make the following changes:

Table 38: Modifying the cluster-config.xml for configuring a Message Server Cluster

Tag with original value	Change value to
<name>default</name>	<name>jmsc1</name> <!--Name of the cluster.-->
<computer-name>127.0.0.1</computer-name>	<computer-name>127.0.0.1</computer-name> Note: You can change the IP of your system here.
<naming-port>9191</naming-port>	<naming-port>9191</naming-port> Note: You can change the port value to the port being used.
<node-type>jms-cluster</node-type>	<node-type>jms-cluster</node-type>

Note: *The number of nodes created in the xml must be equal to the number of <new folders> created under <install_dir>/server/nodes.*

Setting up an HA Message Server Cluster

To set up a HA Message Server cluster node:

1 Create the Node Directory and copy the configuration files required for a Message Server node as mentioned in Table 2.

2 Add cluster declaration in `server_config.xml`. Make sure the JMS Service is also enabled and Web and EJB are disabled.

```
<service name="WebContainer" enabled="false"
    class-name="com.pramati.web.WebServer">
<service name="EJBContainer" enabled="false"
    class-name="com.pramati.ejb.EJBContainer">
<service name="JMSService" enabled="true"
    class-name="com.pramati.jms.server.PramatiMessageService">
```

3 Enable Cluster service.

```
<service name="ClusterService" enabled="true"
    class-name="com.pramati.cluster.PramatiClusterService">
```

4 Setup persistence.

5 Setup cluster topology in `cluster-config.xml`.

Note: *Make sure the node types mentioned in the node definitions in cluster-config.xml are only of the type jms-cluster.*

A sample node configuration for a Message Server cluster:

```
<node>
    <name>node</name>
    <computer-name>jms1</computer-name>
    <naming-port>2099</naming-port>
    <node-type>jms-cluster</node-type>
    <node-id>001</node-id>
</node>
```

The above snippet is from `cluster-config.xml`.

Note: *A Message Server Cluster cannot have a Web/EJB node. Hence make sure that the Web/EJB container services are disabled for all the Message Server nodes.*

Starting Message Server node from Command line

To run any command using the command line, you must first run the `setup.bat` or `setup.sh` in the /bin directory. Enter the following at the command prompt to start the Message Server node:

```
runserver -node [node_name]
```

To connect to the node, type http://<IP>:<naming port>/admin. The Console page appears.

Configuring JMS Adapters in J2EE Servers 22

JMS Adapters are setup in a J2EE Server instance determining the connectivity to a Message Server (Embedded, Standalone or HA-Message Server). JMS Adapters can be configured in `resource-config.xml` file of the J2EE Server (Standalone or cluster-node).

In case of non-embedded Message Server, that is when the JMS Service is not running in the same VM as the EJBService, specify:
`com.pramati.naming.client.PramatiClientContextFactory` for `java.naming.factory.initial`. Also, specify the IP and port of the VM where the Message Server is running in `resource-config.xml`. In case of HA Message Server, provide the comma separated list of URLs of HA Message Server nodes.

Sample Configuration for JMS Adapters

Sample configuration for the local JMS Adapter is shown below. The sample configuration is valid only for a Message Server running in embedded mode:

```
<jms-adapters>
     <jms-adapter name="default" description="nodesc" interface-
     class="com.pramati.jms.client.JMSProviderImpl">
     <properties>
            <property name="java.naming.factory.initial"
            value="com.pramati.naming.PramatiLocalContextFactory"/>
            <property name="java.naming.provider.url" value="rmi://
            localhost:9191"/>
     </properties>
     </jms-adapter>
</jms-adapters>
```

Sample configuration for the external JMS Adapter (non-HA) is shown below. The sample configuration is valid only for a Message Server not running in embedded mode. `jmshost` is the host where the Standalone Message Server is running on Port 2099:

```
<jms-adapters>
     <jms-adapter name="default" description="nodesc" interface-
     class="com.pramati.jms.client.JMSProviderImpl">
     <properties>
        <property name="java.naming.factory.initial" value="com.pra-
       mati.naming.PramatiClientContextFactory"/>
     <  property name="java.naming.provider.url" value="rmi://
       jmshost:2099"/>
```

```
      </properties>
      </jms-adapter>
</jms-adapters>
```

Sample configuration for the external JMS Adapter (HA) is shown below. The sample configuration is valid only for an HA-Message Server cluster. *jms2, jms1* are part of HA Message Server configuration.

```
<jms-adapters>
      <jms-adapter name="default" description="nodesc" interface-
      class="com.pramati.jms.client.JMSProviderImpl">
      <properties>
         <property name="java.naming.factory.initial" value="com.pramati.nam-
         ing.PramatiClientContextFactory"/>
         <property name="java.naming.provider.url" value="rmi://
         jms2:9191,rmi://jms1:9191"/>
      </properties>
      </jms-adapter>
</jms-adapters>
```

Note: *For HA Message Server node or a Standalone Message Server node, the resource-config.xml should not contain any JMS Adapters.*

Configuring Pramati Server For Firewalls 23

In typical deployment scenarios, servers are located behind firewalls and any RMI transport layer that attempts to open direct sockets to hosts on the Internet will not be allowed to do so. The default RMI transport provides two alternate HTTP–based mechanisms that enable a client to invoke a method on a remote object residing across the firewall. These are:

- HTTP tunneling when the RMI call is within the HTTP protocol
- Using the default socket factory

This section contains information about:

- HTTP tunneling (Port Filtering firewalls)
- Configuring the client
- Configuring the RMI Server
- Working with NAT firewalls
- Default Socket Factory
- Configuring Server for firewalls
- Hosting Scenarios

HTTP tunneling (Port Filtering firewalls)

To get across a firewall, the transport layer embeds an RMI call within the firewall–trusted HTTP protocol. The RMI call data is sent outside as the body of an HTTP POST request, and the information returned is sent back in the body of the HTTP response.

The transport layer formulates the POST request in two ways.

6. If the firewall proxy forwards an HTTP request to an arbitrary port on the host machine, then it is forwarded directly to the port on which the RMI service is listening. The default RMI transport layer listens on a server socket that understands and decodes RMI calls inside POST requests.
7. If the firewall proxy forwards HTTP requests to well-known HTTP ports, then it is forwarded to the HTTP server listening on port 80 of the host machine, and a CGI script forwards the request to the target RMI server port on the same machine.

Configuring client

While there is no special configuration necessary to enable the client to send RMI calls through a firewall, the client can disable the packaging of RMI calls as HTTP requests. To do this, set the boolean value of `java.rmi.server.disableHttp` property to '`true`.'

Configuring RMI Server

A client outside the domain of server host attempting to invoke methods on remote objects requires to first locate Server. The remote references that Server exports must contain the fully qualified name of Server host. Some firewall proxies do not forward the host name if the host name is specified as the IP address of the host.

On some platforms and network environments, the fully qualified name of the host may or may not be available with Server VM. If so, the fully qualified name of the host must be specified with the property `java.rmi.server.hostname` while starting Server.

For example, to start the Server class com.pramati.Server on the machine docs.pramati.com, run

```
java -Djava.rmi.server.hostname=docs.pramati.com com.pramati.Server
```

If Server does not support RMI clients that are behind firewalls preventing them from forwarding to arbitrary ports, then use this configuration:

- HTTP server on Port 80
- A CGI script located at the aliased URL path: `/cgi-bin/java-rmi.cgi`
- A servlet mapped to the URL path: `/cgi-bin/java-rmi.cgi`, which redirects the call to the RMI Server.

This CGI script:

- invokes the local Java interpreter to execute a class internal to the transport layer that forwards the request to the appropriate RMI Server port, and
- defines properties in the VM using the names and values of the CGI 1.0-defined environment variables.

A sample script is supplied in the RMI distribution for Unix, Solaris and Windows Operating Systems. The script must specify the complete path to the Java interpreter on Server.

Working with NAT firewalls

Network Address Translation (NAT) or IP masquerading implies that the IP of Server host is mapped to a global IP outside the firewall, so that any request arriving at the global IP is transparently directed to the local IP.

For Server to receive calls from the global IP across the firewall, point `java.rmi.server.hostname` to the global IP while starting Server.

For example, if the user's global IP is 192.12.1.143, start Server using:

```
java -Djava.rmi.server.hostname=192.12.1.143 com.pramati.Server
```

Default Socket Factory

The RMI transport layer extends the `java.rmi.server.RMISocketFactory` class to provide a default implementation of a socket factory. The implementation provides resources to client and server sockets, and creates sockets that transparently provide firewall-tunneling mechanism.

Client sockets automatically attempt HTTP connections to hosts that cannot be contacted through a direct socket. Server sockets automatically detect if a newly accepted connection is an HTTP POST request, and if so, returns a socket that will expose only the body of the request to the transport and format its output as an HTTP response.

Client-side sockets having this default behavior are provided by the java.rmi.server.RMISocketFactory.createSocket method. Server-side sockets having this default behavior are provided by the java.rmi.server.

Configuring Server for firewalls

Administrators must consider the following points while using Server with firewalls:

- The naming service port will always have to be open.
- HTTP Tunneling drastically reduces performance of network communication and must be avoided if possible.

To configure Pramati Server or a Cluster node to run behind a firewall:

1. Open the HTTP port. In a typical firewall configuration, this is the default port that is open.
2. Open the Naming service port. This is mandatory if Naming service lookups have to occur from the client-side.

   If HTTP tunneling is used, no other ports need to be open.
3. Open the port on which remote objects are to be exported if HTTP tunneling is not used.

   This port is already configured as part of Server. See the tag <export-port> under <server-nodes> in Server configuration file. The value is by default zero, when it exports remote objects on random ports. Specifying an unused port here exports all remote objects onto this port. Hence to enable the client to talk to remote objects behind a firewall, this port must be open.
4. To enable dynamic downloading of stubs to clients across firewalls the class file server port must also be open. This port can be specified in `server-config.xml` under <class-file-server-port>. By default it is 5020.

   Note: *If dynamic downloading of the EJB stubs is not required, then there is no need to open this port across the firewall.*
5. The configuration file of Server must have the global IP of the node specified in the tag <host-ip>. When starting Server, the following command line argument is given to the JVM:

   ```
   java -Djava.rmi.server.hostname = <GLOBALIP> com.pramati.Server
   ```

Hosting Scenarios

Some of the typical client-server hosting scenarios are:

All installations on the host machine

Nothing exists in the De-militarized Zone (DMZ) behind the firewall. Pramati Server is on the host machine. Pramati Web Container handles all HTTP requests.

Pramati Web Container in DMZ

Pramati Web Container and EJB Container are configured in two different zones: the Web Container is in the DMZ and handles all the HTTP requests. The EJB Container is on the host machine and serves EJBs.

Third-party HTTP server configured with Pramati Server

An existing HTTP server is configured with Pramati Server. While Pramati Server serves EJBs and JSP pages (dynamic content), HTTP server serves static content. In this scenario, Administrators must install Pramati WebGate, the HTTP server plug-in facility, for Pramati Server to work with other HTTP servers.

Third-party HTTP server and Servlet engine

HTTP server with Servlet engine serves HTTP requests from the DMZ, while Pramati EJB Container serves EJBs from the host machine. In this scenario, administrators must place Pramati Server-specific `client.jar` in the classpath of the Servlet engine.

Hosting applications on multiple independent Servers is the simplest of the Server configurations. Each Server is installed on the local host or a remote host. In this configuration, you deploy the complete application on each Server machine. To do this, you use the Deploy Tool in the Console to repeatedly deploy the application on each running Server.

When you deploy an application this way, the session is neither backed up nor shared between two or more Servers. This implies that load balancing and fail over is not available to the application.

To deploy applications across multiple Pramati Servers while sharing a common external HTTP server, install Pramati WebGate, the HTTP Server plugin that ships with the Server.

Upgrading Pramati Server License

This chapter discusses how to obtain license information and upgrade license of the server.

Reading the Current License Information

To obtain license information, the command `license_details` can be used:

```
j2eeadmin@default> license_details

License Information:
    Licensee            : XXXXXX
    Installation Date   : 21 Jan 2004
    Expiry Date         : 21 Jan 2006
    License Type        : Development
    Product             : Pramati Server 4.1
    Start Date          : --
    Available Time      : --
j2eeadmin@default>
```

Upgrading license

In a production environment, once the key expires, the current license can be upgraded from Server J2EE Shell using the following command:

```
upgrade_license <new_key>
```

Server asks for some information that are shown in the following sample:

```
Please Enter the following values required by the command:
Note: Fields followed by * are mandatory
      You can press 'Enter' for default values shown in [ ]
      Typing 'q' and pressing enter at any point will cancel the cur-
rent command
j2eeadmin@default>  - LicenseKey * XXXXXXXXXXXX
License upgraded successfully
Upgraded license successfully. New license details are
License Information:
    Licensee            : XXXXXX
    Installation Date   : 21 Jan 2004
    Expiry Date         : 21 Jan 2006
    License Type        : Development
```

```
    Product             : Pramati Server 4.1
    Start Date          : --
    Available Time      : --
j2eeadmin@default>
```

In this sample, the Xs represent upgraded license key.

Upgrading License at Server Startup

License can also be upgraded at server startup by using the -upgradelicense option as shown below.

```
> runserver -upgradelicense XXXXXXXXXXXXXXXX
C:\PServer41_1127\server\bin>java.exe -Din-
stall.root=C:\PServer41_1127\server -Djava.security.pol-
icy=C:\PServer41_1127\s
pramati\pramati.java.policy -Djava.security.auth.pol-
icy=C:\PServer41_1127\server\lib\pramati\pramati.jaas.policy -Dorg.x
ser=org.xml.sax.helpers.XMLReaderAdapter -
Dorg.xml.sax.driver=org.apache.crimson.parser.XMLReaderImpl
com.pramati.Server -upgradelicense 08xEW0AJDCsCFEQ6

License upgraded successfully
```

Migrating from Pramati Server 3.0 to Pramati Server 4.1

Pramati Server 4.1 provides a number of new features and enhanced functionality. There have been many changes in the internal architecture with respect to Pramati Server 3.0. Specifically, configuration of the Server has been improved and it provides various configuration files for configuring certain Server features.

However, the deployment of applications has not changed much. In this chapter, we will discuss the following areas that will help in migrating from Server version 3.0 to 4.1:

- Command line changes for starting various servers
- Offline preparation in creating client jars
- Remote Shell
- Command Line Operations
- Shell Commands
- System Properties
- Starting Remote Shell
- Startup Options
- Server Configuration XMLs
- Handling RARs

Starting the Servers

The following sections list the difference in the procedure to start various Server for versions 3.0 and 4.1.

Starting J2EE Server

Table 1: Commands for starting J2EE Server

3.0	4.1
com.pramati.J2eeServer	com.pramati.Server -node <j2ee_node_name> (J2EE node is also the default node, with no -node argument)

Starting JMS Server

Table 2: Commands for staring JMS server

3.0	4.1
com.pramti.JMSServer	com.pramati.Server -node <jms_node_name>

Starting Web Load Balancer

Pramati Web Load Balancer has been introduced in version 4.1

Table 3: Commands for starting erb Load balancer

3.0	4.1
Not Available.	com.pramati.Server -node <lb_node_name>

Offline preparation of client.jar

This feature has been introduced in Server 4.1.

Table 4: Command to prepare a client JAR

3.0	4.1
Not Available	com.pramti.ejb.gen.ClientJar <path of app>

RemoteShell

Table 5: Command to connct to Remote Shell

3.0	4.1
com.pramati.RemoteShell	com.pramati.RemoteShell

Command Line Operations

Table 6: Other commands for Server

3.0	4.1
- shell	-shell is default. Use -noshell for disabling shell
-config %APPSERVER_HOME%\nodes\StandAlone\<nodeName>\config\server-config.xml	-node <nodeName>
-nojms	This option has been moved to server-config.xml.[<service name="JMSService" enable="false">]
-keep	This has been moved to deploy-config.xml [<retain-generated-java-files>true<</retain-generated-java-files>]
	com.pramati.Launcher (Server launcher)

Table 6: Other commands for Server

3.0	4.1
	com.pramati.NodeCreator (creates nodes for a Cluster)

Shell Commands

Table 7: Shell commands

Shell	3.0	4.1
j2eeshell	shutdown	shutdown, shutdown <time_to_wait> [after this time system.exit will be called]
ressh	adddatasource	add_datasource
	removedatasource	remove_datasource
	addjmsserver	add_jmsadapter
	removejmsserver	remove_jmsadapter
	addconnfactory	add_jca_cf
	removeconnfactory	remove_jca_cf
	addmailresouce	add_mailresource
	removemailresource	remove_mailresource
		add_jdbc_cf
		remove_jdbc_cf
		add_urlresource
		remove_urlresource

System Properties

Table 8: System Properties

3.0	4.1
-Dcom.pramati.ejb.nolockbmp=true	-Dcom.pramati.ejb.nolockbmp=true
-Dcom.pramati.useMultipleClassLoaders=true	Moved to deploy-config.xml [<classloader-for-multiple-apps enabled="false">]
	-Dcom.pramati.naming.uselocalcontextfactory=true (to use local context factory for intra VM calls)
	-Dcom.pramati.ejb.poolwaittimeoutseconds=20 (maximum time a request would wait for an EJB instance)
	-Dcom.pramati.ejb.enablefreepool=true (use free pool for EJBs)
	-Dcom.pramati.ejb.loadfromdbatstart=true (loads bean with transaction)

Table 8: System Properties

3.0	4.1
	-Dcom.pramati.rmi.enablepassbyreference=true (enable pass by reference for intra VM calls)
	-Dcom.pramati.application.preferdeployedclasses=application
	-Dcom.pramati.application.preferdeployedclasses=web-module
	-Drunning_in_tomcat_mode=true (Tomcat spec-deviation tolerance)
	-Dcom.pramati.web.logging=true(enable web logging)

The command list_system_properties from j2eeshell lists all properties with values. Running "help" or "shelp" gets more information in Shell.

Starting RemoteShell

The remote shell for Server can be started using one of the following commands:

- java com.pramati.RemoteShell -ip <ip> -port <port> -realm <realm> -user <user> -password <password>
- java com.pramati.RemoteShell -file <fileName> filecontents connect -ip <ip> -port <port> -realm <realm> -user <user> -password <password>

List of all start up options to start Pramati Server

These startup options apply for both Server versions.

Table 9: Startup options

Option	Purpose
-verbose	runs Server in verbose mode
-startapps	starts all applications deployed
-redirect	redirects standard output
-noshell	does not start the shell, starts by default. This is new in Version 4.1.
-user	user name to start Server
-password	password to start Server
-realm	realm to start Server
-cleanupapps	cleans up corrupt app directories, if any
-traceprops	properties to enable tracing for Server start, argument the trace props file
-testJDBCResources	tests all JDBC resources during Server startup
-nodedir(argument)	directory of Server node, to be used if the default directory is not used
-uploadconfig	relevant only for a cluster node.uploads the configuration of the node to the database.

New Server Configuration XMLs in Version 4.1

The server-config.xml in Version 3.0 has been split up into the following sub-configuration files:

- server-config.xml for configuring the service information.
- individual service config XMLs for configuring individual services.

Note: *All types of Server nodes always run off server-config.xml instead of the earlier separate central configurations. To recall, in Version 3.0 the embedded JMS server had emb-jms-config.xml, the cluster node had cluster-config.xml as the central configuration stores.*

New Service Configurators and other XMLs in Version 4.1

- deploy-config.xml
- jms-config.xml
- ejb-cache.xml
- resource-config.xml
- web-config.xml
- management-config.xml
- dashboard.xml
- history-collector.xml
- alerts-config.xml
- logging-config.xml
- web-cache.xml
- web-lbconfig.xml

New configuration tags in deploy-config.xml

Table 10: Configuration tags in deploy-config XML

New Tag	Purpose
<application-hooks>	for per application hooks
<classloader-for-multiple-apps enabled="false">	for using single classloader for group of applications
<application-properties versioning-enabled="true">	for versioning of applications
<out-of-process-compilation enabled="false">	for out of the process compilation
<auto-deploy>	the path of the auto deploy directory
<startup-hooks>	for Server hooks

Handling of RARs

- connector-config.xml has been removed. The contents are now in pramati-j2ee-server.xml.

- You can now configure test parameters through Shell while creating a JCA connection factory. You can also configure test parameters through pramati-j2ee-server.xml.

Downloading and Installing WebStart

Downloading and Installing WebStart on Windows

Download WebStart from `http://java.sun.com/products/javawebstart/download-windows.html` and save `javaws-1_0_1_02-win-us-rt.exe` on your local machine.

To install:

1. Double click `javaws-1_0_1_02-win-us-rt.exe`. The Installation Wizard appears.
2. Follow the instructions and complete the installation process.

Downloading and Installing WebStart on Solaris and Linux

The installation files for Solaris and Linux platforms are available as a .zip. Extract this file and run `install.sh`.

If you are working on Linux, download WebStart from `http://java.sun.com/products/javawebstart/download-linux.html` and save `javaws-1_0_1_02-linux-int.zip` on your local machine.

If you are working on Solaris, download WebStart from `http://java.sun.com/products/javawebstart/download-solaris.html` and save `javaws-1_0_1_02-solsparc-int.zip` on your local machine.

To install WebStart:

1. Extract the contents of the *.zip file to your local directory.
2. At the shell prompt, run `install.sh`. The Installation Wizard appears.
3. Follow the instructions and complete the installation process.

Server-side Setup of WebStart

For clients to access any application deployed on your server using WebStart, you need to:

1. Modify Server configuration to support the JNLP MIME type.
2. Sign all the JAR files in the application with a trusted certificate.
3. Package your application as a WAR or EAR.
4. Deploy the application on the Server.
5. Create the JNLP file for the application.

Modifying Server Configuration for WebStart

To use WebStart and launch applications, you need to make the following modifications to the server configuration:

1. Open `server-config.xml` located at <install_dir>/nodes/<node_name>/config/ in an editor.
2. Under the web-container tag of the server-config file, add a new MIME type using the following code:

```
<mime-type>
   <name>.jnlp</name>
   <type>application/x-java-jnlp-file</type>
</mime-type>
```

3. If you have masked *.jar files and *.xml files in the server configuration, remove or comment these values in the file.

```
<masked-file-list>
    <masked-file>.jsp</masked-file>
    <masked-file>.java</masked-file>
    <!--<masked-file>.xml</masked-file>-->
    <masked-file>.war</masked-file>
    <!--<masked-file>.jar</masked-file>-->
</masked-file-list>
```

4. Save the file.

The server configuration is now modified and you can now use WebStart.

Pramati Server Communication Ports and Protocols

Enterprise networks face risks from unauthorized and malignant users who try to gain access to the server information in various ways. Application servers allow administrators to secure some or all communication ports on the server, over which sensitive information such as credit card numbers, may be transported. Internally, server management services use both secure and non-secure ports, some of which may be configured.

The following tables and accompanying figures detail the various ports on Pramati Server, the default values, whether they can be configured, and the protocols used to transport information. The information here is classified into:

- Pramati Server Administration Service
- Pramati Server Standalone Node
- Pramati Server Cluster Node

Ports used by Server Administration Service

Table 11: Communication ports and protocols used by Pramati Server Administration Service

Service Type	Default Port number	Configurable	Description
HTTP	4748.	Yes	Used by browser (web-based management client) connecting.
HTTPS	Not enabled. If enabled, the default is 443.	Yes	Can be used by browsers for secured communications.
Naming	4141.	No	Used by other Administration Services.
Export	Randomly allotted at runtime.	No	Used by Java clients for communicating with Server.
Discovery	4142.	No	UDP-based port used by Administration Services to discover each other on the network.
Internal Communication	4747.	No	Used internally by Administration Service and need not be open to external access.

Ports used by Pramati Server Standalone Node

Table 12: Communication ports and protocols used by Pramati Server Standalone Node

Service Type	Default Port number	Configurable	Description
HTTP	8181.	Yes	Used by browsers.
HTTPS	Not enabled. If enabled, the default is 443.	Yes	Can be used by browsers for secured communications.
Naming	9191.	Yes	Used by Java Clients (EJB/JMS/Admin) for looking up.
Class File Service	5020.	Yes	Used by Java clients to dynamically download stubs.
Export	Randomly allotted at runtime.	Yes	Used by Java clients for communicating with Server.
SSL Export	Not enabled. If enabled, randomly allotted at runtime.	Yes	Can be used by Java clients for secured communications.

Ports used by Pramati Server Cluster Node

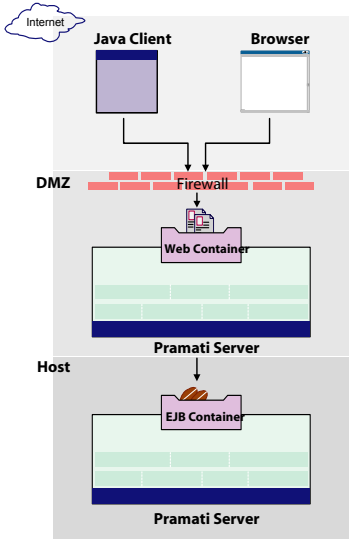
Table 13: Communication ports and protocols used by Pramati Server Cluster Node

Service Type	Default Port number	Configurable	Description
HTTP	8181.	Yes	Used by browsers.
HTTPS	Not enabled. If enabled, the default is 443.	Yes	Can be used by browsers for secured communications.
Naming	9191.	No	Used by Java Clients (EJB/JMS/Admin) for looking up.
Class File Service	5020.	Yes	Used by Java clients to dynamically download stubs.
Export	Randomly allotted at runtime	Yes	Used by Java clients for communicating with Server.
SSL Export	Not enabled. If enabled, randomly allotted at runtime.	Yes	Can be used by Java clients for secured communications.
Node Status Pulse	Randomly allotted at runtime.	No	Is used by nodes to exchange status (up/down) information.

The various ways in which Pramati Server can be installed are:

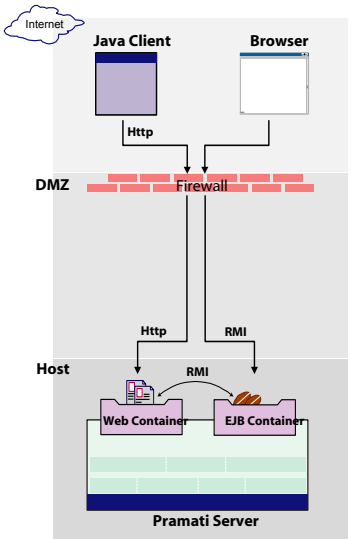
- All installations in the host machine
- Third party HTTP server replacing Pramati HTTP server
- Third party HTTP server + Servlet engine
- Independent Pramati Servers on the network

Nothing exists in the De-Militarized Zone (DMZ) behind the firewall. Pramati Server is on the host machine. Pramati Web Server handles all HTTP requests.



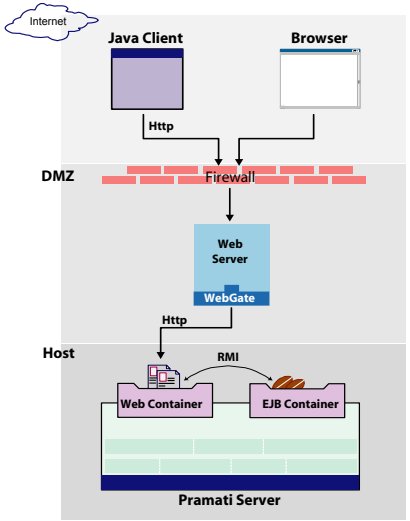
Pramati Web Server in DMZ

Pramati Web Server and EJB Server are configured in two different zones. The Web Server is in the DMZ and handles all the HTTP requests. The EJB Server is on the host machine and is the runtime environment for EJBs.



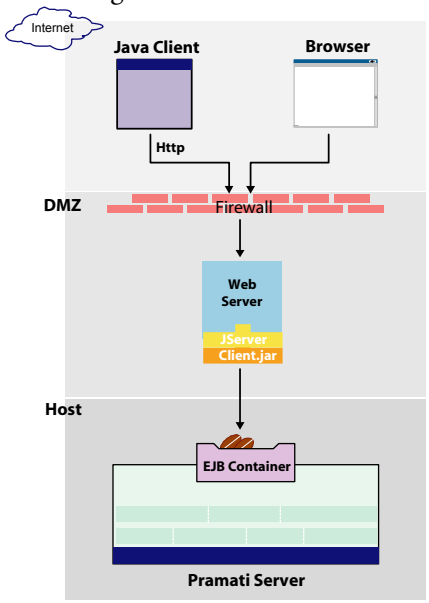
Third-party HTTP server replaces Pramati HTTP Server

Third-party HTTP servers, such as Apache HTTP Server or Microsoft IISTM, is configured with Pramati Server. While Pramati Server serves dynamic content through JSP pages and EJBs, the HTTP server serves static HTML content. In this scenario, Pramati WebGate, and the HTTP server plug-in must be installed on the third-party server.



Third-party HTTP server + servlet engine

Third-party HTTP server with servlet engine serves HTTP requests from the DMZ, while Pramati EJB Server serves EJBs from the host machine. In this scenario, Pramati Server-specific client.jar must be in the classpath of the Servlet engine.



Independent Pramati Servers on the network

Hosting applications on multiple independent servers is the simplest of all Server configurations. Suppose, Pramati Server is installed on the local host and remote host. In this scenario, you deploy the complete application on each Server using the Deploy Tool.

Alternatively, you can prepare the application using the Deploy Tool, and distribute the prepared archive (.par) to multiple Server instances using Pramati Server Management Console. However, sessions are neither backed up nor shared among these Servers, and so load balancing and failover are not available to such an application.

Configure the multiple Pramati Servers as Pramati Server Cluster Nodes for load balancing and failover of the application.

To deploy applications across multiple Pramati Servers that are sharing a common external HTTP server, install Pramati WebGate, the third-party HTTP server plug-in.

Adding a Load Balancer node 45
Adding a User and Changing Password 31
Adding node(s) to Message Server Cluster 66
Adding User 31
Adding Web Load Balancer using Administration Service 43
Advanced Configuration 94
All installations on the host machine 108
Browser 2
Changing Password 32
Changing password for User 33
Cluster Configuration using NodeCreator 85
Cluster Definition 89
Command Line Operations 112
Configurable Server Components 25
Configuration File Dependency 76
Configuration Layouts 121
Configuring Classpath Parameters 24
Configuring client 105
Configuring Components by Modifying Configuration XMLs 27
Configuring Components using Console 27
Configuring Components using Pramati Server Administration Service 27
Configuring J2EE Cluster 51
Configuring J2EE Cluster Using Server Administration Service 51
Configuring J2EE Server Cluster using NodeCreator 86
Configuring J2EE Server Cluster using NodeCreator 97
Configuring J2EE Server Cluster using XMLs 96
Configuring JMS Adapters in J2EE Servers 103
Configuring JMS Persistence Framework 59
Configuring JVM Parameters 23
Configuring Large Message Handling 59
Configuring Load Balancer 46
Configuring Message Server as a Standalone Node 55
Configuring Message Server Cluster 65
Configuring Message Server Cluster Using Server Admin Service 65
Configuring Message Server Cluster Using XML 99

Configuring Message Server Cluster using XML 99
Configuring more than one Pramati Server Nodes 22
Configuring Pramati Server For Firewalls 105
Configuring RMI Server 106
Configuring Server for firewalls 107
Configuring Standalone Message Server 55
Configuring Standalone Message Server 55
Configuring Standalone Pramati Server Using Admin Service 37
Configuring Standalone Server 37
Configuring Standalone Server 39
Configuring the Message Server for Loaded Queues 57
Configuring Web Load Balancer Using XML 49
Connecting to Cluster 53
Connecting to Cluster 67
Connecting to Load Balancer 48
Connecting to Standalone Message Server 56
Connecting to Standalone Server 40
Contact Information 1
Creating a Cluster Instance 90
Creating a Standalone Server 76
Creating and Configuring Pramati Server Nodes Using XML 71
Creating LB Node using NodeCreator 87
Creating Pramati Server Clusters Using XMLs 89
Creating Standalone Message Server 55
Creating Standalone Message Server 79
Creating Standalone Server 38
Customizing Default Parameters 22
Customizing Server Footprint 79
Database Requirements 2
Default Socket Factory 107
Deleting Cluster 54
Deleting Cluster 69
Deleting User 32
Deploying an application on Cluster node 53
Deploying an application on Cluster node 97
Document Conventions 1

Downloading and Installing WebStart 117
Downloading and Installing WebStart on Solaris and Linux 117
Downloading and Installing WebStart on Windows 117
Effects of deploying an application on Cluster node 54
Enabling embedded Message Server in a Standalone Server 78
Getting Server Install File 5
Handling of RARs 115
Hardware Requirements 1
Hosting Scenarios 108
How does Server Administration Service work 36
HTTP tunneling (Port Filtering firewalls) 105
Independent Pramati Servers on the network 123
Installing in non-GUI mode from command line 13
Installing Java Wrapper 21
Installing Pramati Server on Unix-based Platforms 9
Installing Pramati Server on Windows 13
Introduction to Pramati Server Administration Service 35
J2EE Cluster Node Types 93
Java Client 27
JDK 2
Key terminology 71
List of all start up options to start Pramati Server 114
Load Balancing Scenarios 43
Location of the extracted application 54
Managing Cluster Nodes 69
Migrating from Pramati Server 3.0 to Pramati Server 4.1 111
Modifying Cluster 52
Modifying Message Server Cluster configuration 68
Modifying Server Configuration for WebStart 118
New configuration tags in deploy-config.xml 115
New Server Configuration XMLs in Version 4.1 115
New Service Configurators and other XMLs in Version 4.1 115
Offline preparation of client.jar 112
Operating System Requirements 1
Overview 71
Ports used by Pramati Server Cluster Node 120

Ports used by Pramati Server Standalone Node 120
Ports used by Server Administration Service 119
Pramati Embedded Message Server 26
Pramati Server Administration Service 27
Pramati Server Cluster BOTH node 26
Pramati Server Cluster EJB node 26
Pramati Server Cluster node 25
Pramati Server Cluster Web node 26
Pramati Server Communication Ports and Protocols 119
Pramati Server Components 25
Pramati Server configuration files 72
Pramati Server instances that can be configured 73
Pramati Server Load Balancer 26
Pramati Server Management Console 27
Pramati Server Standalone Message Server node 26
Pramati Server Standalone node 25
Pramati Web Container in DMZ 108
Pramati Web Server in DMZ 122
Preface 1
Procedure for creating Pramati Server Instances 75
Reading the Current License Information 109
Receiving License Key 6
RemoteShell 112
Removing a Load Balancer 48
Removing Standalone Message Server 57
Removing Standalone Server 40
Restricting services in a Standalone J2EE Server 78
Role of Administration Service 35
Running client for Message Server Cluster 69
Sample Configuration for JMS Adapters 103
Server Cluster Node configurations derived from an instance 73
Server Instance Directory Structure 74
Server-side Setup of WebStart 117
Setting up a J2EE Cluster 92
Setting up an HA Message Server Cluster 100
Setting up EJB Persistence 95

Setting up Pramati Server as a Service 19
Setting up Pramati Server as a Service on Windows 19
Setting up Pramati Server as a Unix Daemon 20
Setting up Web Replication and Persistence 94
Shell Commands 113
Silent Installation of Pramati Server Using XML 17
Software Requirements 2
Standalone Server configurations derived from an instance 73
Starting a Server Node Using Command line 63
Starting Cluster 53
Starting J2EE Server 111
Starting JMS Server 112
Starting Load Balancer 47
Starting Message Server Cluster 67
Starting Message Server node from Command line 101
Starting RemoteShell 114
Starting Server Administration Service 36
Starting Standalone Message Server 56
Starting Standalone Message Server using Command line 58
Starting Standalone Server 39
Starting the Default Server on Unix 37
Starting the Default Server on Windows 37
Starting the Servers 111
Starting Web Load Balancer 112
Stopping Cluster 54
Stopping Cluster 68
Stopping Load Balancer 48
Stopping Message Server node 68
Stopping Standalone Message Server 57
Stopping Standalone Server 40
Structure of Nodes Directory 74
System Properties 113
System Requirements 1
System Requirements for Installing Pramati Server 1
The installation directory structure 74
Third Party Integration with Apache/IIS 27

Third-party HTTP server + servlet engine 123
Third-party HTTP server and Servlet engine 108
Third-party HTTP server configured with Pramati Server 108
Third-party HTTP server replaces Pramati HTTP Server 122
Troubleshooting Tips 24
Understanding Message Server Configuration 59
Understanding Pramati Server nodes 71
Updating cluster configuration using XML (when DB is used for configuration) 95
Upgrading license 109
Upgrading License 6
Upgrading License at Server Startup 110
Upgrading License at Server Startup 6
Upgrading License using Command Shell 6
Upgrading Pramati Server License 109
Uploading Configuration to DB when DB is used for Configuration Persistence 90
Usage Objective 85
Using - configuration flag for custom node configuration 84
Using Default Server 37
Using JDK 3
Using JRE 3
Using Loaded Send Queues 57
Using NodeCreator with Command line 81
Using NodeCreator with XMLs 82
Using Pramati Server NodeCreator 81
Using Security option 32
Using User Settings 32
Verifying and Using JDK and JRE 2
Verifying installation of Deploy Tool 14
Verifying installation of Deploy Tool on Unix-based platforms 9
Verifying installation of Message Server 15
Verifying installation of Message Server on Unix-based platforms 10
Verifying Server installation 14
Verifying Server installation on Unix-based platforms 9
Viewing details of added Standalone Server 39

Viewing details of Standalone Message Server 56
Viewing Load Balancer details 47
Viewing Message Server node details 68
Working with NAT firewalls 106

C
Changing Password 32
Configuring
    Server 107
Configuring Standalone Server 39
D
Default Socket Factory 107
H
Handling of RARs 115
HTTP Tunneling 105
L
license details 109
Load Balancing Scenarios 43
O
Obtaining License Information 109
Offline preparation of client.jar 112
P
Pramati Server WebGate 45
Pramati Technologies 37, 49
S
Server Version 36
Starting RemoteShell 114
U
Upgrading License Key 109
Upgrading the License Key 109
Users
    Add 31
    Delete 32
W
Working with NAT Firewalls 106